# NSFORT--A NONSTANDARD FORTRAN LANGUAGE TRANSLATOR

by

David Ross Little

# United States Naval Postgraduate School


# THESIS

NSFORT--A NONSTANDARD FORTRAN LANGUAGE TRANSLATOR

by

David Ross Little

September 1970

# NSFORT--A Nonstandard FORTRAN Language Translator

by

David Ross Little
Captain, United States Marine Corps
B.S., United States Military Academy, 1963

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1970

ABSTRACT

This paper presents a computer language translator which allows the IBM FORTRAN G (or higher level) user to solve directly problems in which the variables may be n-tuples and/or the FORTRAN arithmetic and relational operations may be defined by the user. The translator, called the NSFORT (for NonStandard FORTRAN) translator, will (1) decompose all expressions to a series of binary arithmetic operations, relational operations, or user defined functions; (2) generate CALL statements to user supplied subprograms to perform the above operations; and (3) produce a new source program that is in every respect acceptable to the FORTRAN compiler. While using the NSFORT translator the user has virtually unrestricted use of FORTRAN. The translator's applications include n-precision arithmetic, vector and matrix operations, numerically evaluated analytic derivatives, interval arithmetic, and others. The paper describes completely the use and operation of the translator, provides examples, indicates applications, and discusses programming techniques.

TABLE OF CONTENTS


I. INTRODUCTION ---------- 5

II. TRANSLATOR DESCRIPTION AND OPERATION ---------- 11

III. PROBLEMS AND TECHNIQUES ---------- 18

APPENDIX A: NSFORT TRANSLATOR PROGRAM ---------- 22

APPENDIX B: JOB CONTROL LANGUAGE USED WITH THE NSFORT TRANSLATOR ---------- 36

APPENDIX C: NSFORT TRANSLATOR PROGRAM DESCRIPTION ---- 41

APPENDIX D: SYNTAX RULES FOR NSFORT STATEMENTS ------- 106

APPENDIX E: SUMMARY OF NSFORT CODES ---------- 109

APPENDIX F: TRANSLATOR ACTION FOR SPECIFIC FORTRAN AND NONSTANDARD FORTRAN STATEMENTS ---------- 112

APPENDIX G: RESULT TYPE OF MIXED TYPE OPERATIONS ----- 128

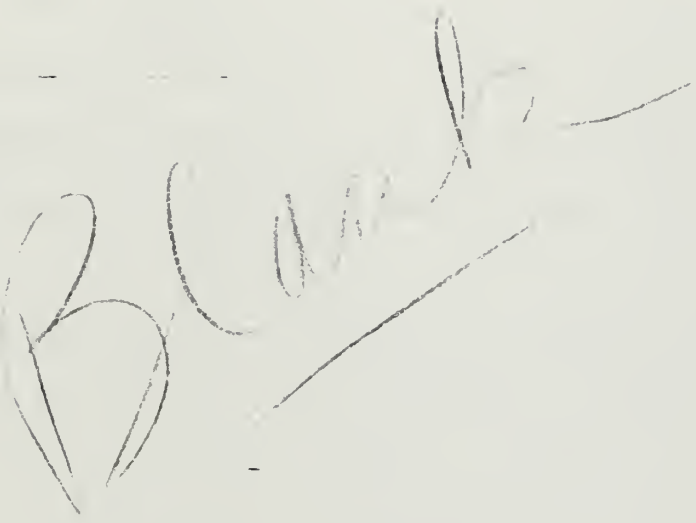
APPENDIX H: DESIGN LIMITATIONS AND RESTRICTIONS ------ 129

APPENDIX I: NSFORT DIAGNOSTIC MESSAGES ---------- 135

LIST OF REFERENCES ---------- 152

INITIAL DISTRIBUTION LIST ---------- 153

FORM DD 1473 ---------- 155



# I. INTRODUCTION

The FORTRAN IV programming language allows the user to evaluate algebraic expressions when the variables are real, complex, or logical scalers, and when the operations are any of the standard FORTRAN* set of arithmetic, relational, or logical operators. The user has some flexibility with FORTRAN and user defined functions as long as the result is also real, complex, or logical. FORTRAN lacks the capability to solve directly a problem in which the variables are n-tuples and/or the operations are not part of the standard FORTRAN set.

This paper presents a language translator (or precompiler), which allows the user to define a nonstandard variable type (as opposed to COMPLEX, INTEGER, LOGICAL, or REAL). As a consequence, the arithmetic and relational operators become undefined in the usual sense. Thus, this translator allows the user to evaluate statements such as A = B $\oplus$ C where A, B, and C may be scalers or n-tuples and $\oplus$ may be defined any way the user wishes (see Chapter II). This translator will be referred to as the NSFORT translator (for NonStandard FORTRAN).

As an example, assume FORTRAN IV did not have the capability to handle complex operations and the user wished

---

*Whenever the words "FORTRAN," "FORTRAN IV," or any of its variations are used herein, that implication refers to IBM FORTRAN G-level or higher (i.e. H, 44PS, and TSS).

to find the product of two complex numbers. Using the NSFORT translator the user could write the following program.

```
      NONSTANDARD1*2 A,B,C
      C=A*B
      END
```

After translation the program would be the following.

```
C
C
C        GENERATED FORTRAN SOURCE PROGRAM
C
C
      REAL*4 T$A(2,1)
      REAL*4 A(2),B(2),C(2)
      CALL N$AAMA (T$A(1,1), A(1), B(1))
      CALL N$AA   (C(1), T$A(1,1))
      END
```

By supplying the following two subprograms the user is able to obtain the desired result.

```
      SUBROUTINE N$AAMA (X,A,B)
      REAL*4 X(2),A(2),B(2)
      X(1)=A(1)*B(1)-A(2)*B(2)
      X(2)=A(1)*B(2)+A(2)*B(1)
      RETURN
      END

      SUBROUTINE N$AA (C,X)
      REAL*4 C(2),X(2)
      C(1)=X(1)
      C(2)=X(2)
      RETURN
      END
```

As a second example take the problem of evaluating a vector (or cross) product of two 3 element vectors. The user can write the NSFORT program and the two subprograms below.

```
      NONSTANDARD1*3 A,B,C
      C=A*B
      END

      SUBROUTINE N$AAMA (X,A,B)
      REAL*4 X(3),A(3),B(3)
      X(1)=A(2)*B(3)-B(2)*A(3)
      X(2)=A(3)*B(1)-B(3)*A(1)
      X(3)=A(1)*B(2)-B(1)*A(2)
      RETURN
      END

      SUBROUTINE N$AA (C,X)
      REAL*4 C(3),X(3)
      C(1)=X(1)
      C(2)=X(2)
      C(3)=X(3)
      RETURN
      END
```

The translated main program is as follows.

```
C
C
C        GENERATED FORTRAN SOURCE PROGRAM
C
C
      REAL*4 T$A(3,1)
      REAL*4 A(3),B(3),C(3)
      CALL N$AAMA (T$A(1,1), A(1), B(1))
      CALL N$AA   (C(1), T$A(1,1))
      END
```

The codes used in the above examples will be fully explained later but briefly the A's in the subroutine names represent the variable types of the arguments, respectively, and the M is an abbreviation for "multiply." The T$A is a temporary variable with A being the variable type.

It should be noted that for the two almost identical main programs above, two totally different operations can be performed simply by supplying different subroutines.

The NSFORT translator performs another significant task not illustrated in the above two examples. The translator decomposes all expressions containing nonstandard variables into a sequence of elementary operations just as the assignment statements above were decomposed into two elementary operations. (In the case above the two operations were multiplication and simple assignment.) Thus the user has only to supply the subroutines for the several elementary operations used in the program and then any complicated expression using several elementary operations and a mixture of variable types can be evaluated.

Specifically the NSFORT translator will translate a nonstandard FORTRAN program into a normal FORTRAN program acceptable to the FORTRAN compiler. The resulting FORTRAN program will have the following features: (1) All nonstandard variables, arrays, and functions will be mapped into conventional variables, arrays, and functions of determined type and determined dimension. (2) All arithmetic and logical expressions containing nonstandard variables, arrays, etc., will be decomposed into a series of elementary operations. (3) Simple assignment or CALL statements will be written to effect the performing of the series of elementary functions. And (4) modifications as required will be made on most other FORTRAN statements to permit the successful compilation and execution of the program.

Applications for the NSFORT translator include but are not restricted to the following:

(1) n-precision integer or real variable tasks

(2) Complex or quaternion variables

(3) n-dimentional vector or matrix operations

(4) Numerically evaluated analytic derivatives [Refs. 1 and 2]

(5) Interval arithmetic [Ref. 3]

(6) Abstract algebraic operations

It should be acknowledged that the overall concept of the NSFORT translator is not new. It was suggested to the author by Professor Rex H. Shudde, Ph.D., and is based on his use of the TYPE OTHER feature of FORTRAN 63 [Refs. 4 and 5].

This thesis will not be concerned with the writing of specific sets of subroutines for specific applications. (However, the NSFORT translator has been completely tested through final execution with numerical differentiation problems and hence, a set of subroutines for this application in normal precision have been written.) The reader should also keep in mind the general nature of the NSFORT translator and not feel restricted to the applications cited. For example, the number of values in an operand or result is not necessarily greater than one; and the user can redefine the elementary functions as he wishes.

Advantages to be noted as the NSFORT translator is described in detail are (1) generality and flexibility of the translator, (2) minimal restrictions in the use of the FORTRAN IV language, (3) complete accounting of variable

types, (4) generation of nonstandard program listing with diagnostic messages, and (5) complete handling of mixed-type expressions. Disadvantages are (1) some restrictions on programming, (2) the inability to automatically handle input/output functions, and (3) the restriction to IBM FORTRAN G (or higher level) compilers.

## II. TRANSLATOR DESCRIPTION AND OPERATION

The NSFORT translator is written in the PL/I programming language [Refs. 6 and 7] and is presently operational on an IBM 360/67 in the OS configuration. For further details about the translator see Appendix A--NSFORT Translator Program Descriptioon, Appendix B--Job Control Language Used with the NSFORT Translator, and Appendix C--NSFORT Translator Program Listing. All conventional FORTRAN usage is in accordance with Reference 8.

As noted previously, the fundamental task of the user is to specify (implicitly and/or explicitly) certain variables, arrays, and functions as nonstandard. Appendix D--Syntax Rules for NSFORT Statements, describes in detail this procedure. In addition to identifying variables the user also must specify the nonstandard type and the nonstandard size. As described in Appendix D, the nonstandard type determines which FORTRAN variable type the nonstandard variable will be mapped into, and the nonstandard size determines the size of the FORTRAN array that the nonstandard variable will map into. The examples below should clarify the relationships.

| NSFORT | FORTRAN |
|---|---|
| NONSTANDARD1*2 ALPHA | REAL*4 ALPHA(2) |
| NONSTD6*3 BETA,GAMMA(10,10) | COMPLEX*16 BETA(3), GAMMA(3,10,10) |
| NSTD2*3 DELTA(100) | REAL*8 DELTA(3,100) |

IMPLICIT specifications will be discussed later.

Elementary operations to which nonstandard expressions will be decomposed are the following:

Assignment (=)

Arithmetic operations (+, -, *, /, **, unary -, unary +)

Relational operations (.LT., .LE., .EQ., .NE.,.GE., .GT.)

Logical operations (.NOT., .AND., .OR.)

Any non-FORTRAN user function

Note that FORTRAN functions (SIN, COS, EXP, ALOG, etc.) are not included in the above list.

Before proceeding the reader should familiarize himself with Appendix E--Summary of NSFORT Codes. For a complete description of translator action on any statement see Appendix F--Translator Action for Specific FORTRAN and Nonstandard FORTRAN Statements.

A simple example will illustrate the translation of the IMPLICIT statement and of elementary assignment statements.

```
NSFORT:         IMPLICIT NONSTANDARD1*2 (A-C)
          C          INPUT
                  CALL INPUT(A1,B1,A2,B2,A3,B3)
          C          COMPUTE
              1   C1=-A1+B1
              2   C2=(A2+B2)*B2
              3   C3=(A3+B3)/A3**B3
              4   C4=(A3+10.)*(D*E)
              5   C5=F
              6   H=P+Q+R
          C          OUTPUT
                  CALL OUTPUT(C1,C2,C3,C4,C5)
                  STOP
                  END
```

```
FORTRAN: C
         C
         C        GENERATED FORTRAN SOURCE PROGRAM
         C
         C
               IMPLICIT REAL*4 (A-C)
         C          INPUT
               REAL*4 T$1(1), T$A(2,3)
               DIMENSION A1(2),B1(2),A2(2),B2(2),A3(2),B3(2),
              *C1(2),C2(2),C3(2),C4(2),C5(2)
               CALL INPUT (A1(1), B1(1), A2(1), B2(1), A3(1),
              *B3(1))
         C          COMPUTE
             1 CALL N$AAN  (T$A(1,1), A1(1))
               CALL N$AAAA (T$A(1,2), T$A(1,1), B1(1))
               CALL N$AA   (C1(1), T$A(1,2))
             2 CALL N$AAAA (T$A(1,1), A2(1), B2(1))
               CALL N$AAMA (T$A(1,2), T$A(1,1), B2(1))
               CALL N$AA   (C2(1), T$A(1,2))
             3 CALL N$AAAA (T$A(1,1), A3(1), B3(1))
               CALL N$AAEA (T$A(1,2), A3(1), B3(1))
               CALL N$AADA (T$A(1,3), T$A(1,1), T$A(1,2))
               CALL N$AA   (C3(1), T$A(1,3))
             4 CALL N$AAA1 (T$A(1,1), A3(1), 10.)
               T$1(1) = D * E
               CALL N$AAM1 (T$A(1,2), T$A(1,1), T$1(1))
               CALL N$AA   (C4(1), T$A(1,2))
             5 CALL N$A1   (C5(1), F)
             6 H=P+Q+R
               OUTPUT
               CALL OUTPUT (C1(1), C2(1), C3(1), C4(1), C5(1))
               STOP
               END
```

The decomposition process is illustrated in this example for a variety of cases. Note particularly statement 2. First the "addition" A2 and B2 is isolated. This result, T$A(·,1), is then "multiplied" by B2 to obtain T$A(·,2), and finally that result is assigned to C2.

Other points to note in this example are

(1) When IMPLICIT specification is used all variables, so specified, used in the program are properly dimensioned by the translator.

(2) Comments are passed unaltered.

(3) Statement numbers (if used) remain with the first generated statement. (The statement numbers associated with DO loops are handled separately. See Appendix F.)

(4) Temporary variables are properly declared and are reused in subsequent statements, thus temporary storage is minimized.

(5) Arrays in arguments of CALL statements are identified by the first storage location.

(6) Simple arithmetic assignment is a distinct elementary function. This is advantageous in many applications (e.g. it eliminates the possibility of logical errors in statements of the type A = A*B) and it serves to reduce the number of subroutines required for a specific application.

(7) All mixed-type operations are permissible. The variable type of the result is predetermined. See Appendix G--Result Type of Mixed Type Operations.

(8) Statements not containing nonstandard variables are unaltered.

It should be clear at this point that the user's use of FORTRAN must be restricted to some degree. For example, the NSFORT translator uses the "$" symbol and hence the user must limit his use of it; likewise since FORTRAN does not permit in excess of seven subscripts, the user is limited to six on nonstandard variables since the translator adds one. These restrictions are enumerated in Appendix H--Design Limitations and Restrictions. In case of a coding error or,

in some cases such as the violation of a design limitation or restriction that prevents the translator from properly processing a statement, the translator will issue a diagnostic message. This message will appear on the nonstandard FORTRAN program listing immediately below the statement in error. For a list of the error messages and explanations see Appendix I--NSFORT Diagnostic Messages.

The following example will illustrate the translation of the logical IF statement and the use of a nonstandard function.

```
NSFORT:         NONSTANDARD2*3 X(2),A,B,C,SQROOT(*)
              1 CALL INPUT (A,B,C)
              2 IF (B**2-4*A*C.LT.0.) GO TO 10
              3 X(1)=(-B+SQROOT(B**2-4*A*C))/2.*A
              4 X(2)=(-B-SQROOT(B**2-4*A*C))/2.*A
              5 CALL OUTPUT (X(1))
             10 STOP
                END

                NONSTANDARD2*3 FUNCTION SQROOT (A)
                NONSTANDARD2*3 A
                  .
                  .
                  .
                SQROOT=...
                  .
                  .
                  .
                END
```

```
FORTRAN: C
         C
         C        GENERATED FORTRAN SOURCE PROGRAM
         C
         C
               REAL*8 T$B(3,9)
               LOGICAL*1 T$7(1)
               REAL*8 X(3,2),A(3),B(3),C(3)
             1 CALL INPUT (A(1), B(1), C(1))
             2 CALL N$BBE4 (T$B(1,1), B(1), 2)
               CALL N$B4MB (T$B(1,2), 4, A(1))
               CALL N$BBMB (T$B(1,3), T$B(1,2), C(1))
               CALL N$BBSB (T$B(1,4), T$B(1,1), T$B(1,3))
               CALL N$7B11 (T$7(1), T$B(1,4), 0.)
               IF (T$7(1)) GOTO10
             3 CALL N$BBN  (T$B(1,1), B(1))
               CALL N$BBE4 (T$B(1,2), B(1), 2)
               CALL N$B4MB (T$B(1,3), 4, A(1))
               CALL N$BBMB (T$B(1,4), T$B(1,3), C(1))
               CALL N$BBSB (T$B(1,5), T$B(1,2), T$B(1,4))
               CALL $SQR00 (T$B(1,6), T$B(1,5))
               CALL N$BBAB (T$B(1,7), T$B(1,1), T$B(1,6))
               CALL N$BBD1 (T$B(1,8), T$B(1,7), 2.)
               CALL N$BBMB (T$B(1,9), T$B(1,8), A(1))
               CALL N$BB   (X(1,1), T$B(1,9))
             4 CALL N$BBN  (T$B(1,1), B(1))
               CALL N$BBE4 (T$B(1,2), B(1), 2)
               CALL N$B4MB (T$B(1,3), 4, A(1))
               CALL N$BBMB (T$B(1,4), T$B(1,3), C(1))
               CALL N$BBSB (T$B(1,5), T$B(1,2), T$B(1,4))
               CALL $SQR00 (T$B(1,6), T$B(1,5))
               CALL N$BBSB (T$B(1,7), T$B(1,1), T$B(1,6))
               CALL N$BBD1 (T$B(1,8), T$B(1,7), 2.)
               CALL N$BBMB (T$B(1,9), T$B(1,8), A(1))
               CALL N$BB   (X(1,2), T$B(1,9))
             5 CALL OUTPUT (X(1,1))
            10 STOP
               END

         C
         C
         C        GENERATED FORTRAN SOURCE PROGRAM
         C
         C
               SUBROUTINE $SQR00 (SQROOT,A)
               REAL*8 SQROOT(3)
               REAL*8 A(3)
                 :
                 :
               CALL N$BB   (SQROOT(1),...
                 :
                 :
               END
```

It is apparent that FORTRAN functions cannot be used with nonstandard arguments, but the above example illustrates how the user can use analogous nonstandard functions and let the translator automatically make the required adjustments.

No attempt has been made to describe completely the action of the NSFORT translator in the body of this paper but only to illustrate the concept and the power of the translator. The reader is again directed to Appendices F and H for a complete description.

## III. PROBLEMS AND TECHNIQUES

The remainder of this thesis will discuss particular problems encountered in the design of the NSFORT translator and the techniques used to overcome them. This project was accomplished with the extremely powerful PL/I programming language. Extensive use in all aspects of the project was made of the character string handling capability of PL/1, notably the VARIABLE attribute of character strings and the character string functions SUBSTR and INDEX.

Clearly the first major task of the translator is the classification of statements. This is accomplished by recognizing the distinctive pattern of statements. Since blank spaces have no effect (except in literals) in FORTRAN they can be eliminated. Thus the first step is to eliminate blanks but only up to the first delimiter so as to not alter any literal. Assignment statements and statement functions are then identified. FORTRAN statements which are neither assignment statements nor statement functions are identified by their first two characters and then by more characters if the first two are not unique. The next requirement is to remove the literals and replace them by a code; after processing the literals are then reinserted. Removing the literals permits the remaining blanks to be eliminated and reduces each statement to an identifiable pattern.

The largest single procedure in the translator is the one that processes expressions. This procedure is used,

as required, for CALL statements, IF statements, and assignment statements, and is designed to decompose logical as well as arithmetic expressions, and to generate the required CALL statements. The reverse Polish string method that is described by Randell and Russell [Ref. 9] is used. The method is simplified somewhat since subscripts may not contain nonstandard variables and hence a variable with its subscripts may be treated as a single entity. On the other hand, the translator's capability to handle nonstandard functions made the processing of expressions somewhat more intricate.

In generalized compiler processing using the reverse Polish method there are three steps: (1) generation of the reverse Polish string, (2) generation of the triplet list, and (3) reduction of the triplet list by eliminating identical triplets and accounting for connected triplets and commutative operations. In the NSFORT translator the generated CALL statements are analogous to the triplet list. Connected triplets offer no advantage in the NSFORT translator but efficiency can be increased by eliminating identical CALL statements. There is a significant trade-off however since sizable storage space must be available for the triplet list with the possible payoff being speedier execution. Since it is anticipated that the NSFORT translator would be used by proficient programmers, it was believed that reducing the triplet list would offer an insignificant advantage. Therefore the following variation

was used. The translator begins generating the reverse Polish string until the first triplet is complete. At this point the triplet is converted to a CALL statement, which is written on an output file, and the temporary variable containing the result of the triplet replaces the triplet in that fragment of the reverse Polish string. The translator then continues to generate the reverse Polish string until the next triplet is complete.

Determining whether or not particular statements contain nonstandard variables was a problem owing to the possibility of the use of either implicit or explicit specifications. This is a YES or NO determination in the NSFORT translator. The technique used is to first check for the presence of explicitly declared nonstandard variables or functions. If none are present then a check is made for implicitly declared nonstandard variables. If any are found they must be checked against the list of explicitly declared variables to insure that the implicit declaration has not been overridden by an explicit declaration of the specific identifier. If in the process of searching the statement the presence of nonstandard variables is ascertained, then the search process is immediately terminated.

The NSFORT translator treats each subprogram as an independent entity, and is designed to process a series of subprograms in one execution. The only delineation required is the normal END statement. To handle subprograms, certain lists of relevant information must be compiled by the

translator. Essentially, these are a list of explicitly declared variables, a list of implicitly declared letters of the alphabet, a list of dimensioned variables, and a list of nonstandard function names. When an END statement is encountered these lists are erased and the translator is prepared for a new subprogram.

The NSFORT translator was designed and coded to handle legitimate FORTRAN statements on an individual basis. It has some built-in safety features (the diagnostic messages) which were designed to keep the translator from abnormally terminating in the system and to prevent the translator from doing something incorrectly. The translator is not designed to detect FORTRAN syntax errors or statement errors, except to the extent described above. Furthermore, an error in a specification statement which would render meaningless the remainder of the subprogram will not terminate translation of that subprogram.

This translator has been extensively tested but, naturally, it was impossible to foresee and to test it with every possible FORTRAN program. This fact combined with the complexity of the task almost certainly assures that future errors in the translation will be uncovered. These will be corrected as they are brought to the author's attention.

The concept embodied in the NSFORT translator could be extended by allowing the user to use unique symbols instead of the set of FORTRAN operators to specify the elementary operations. In this manner the generated subroutine names could be made unique.

## APPENDIX A

## NSFORT TRANSLATOR PROGRAM DESCRIPTION

The NSFORT translator is written in the PL/I language and consists of twenty-six procedures (subprograms). The procedure MAIN is the controlling program and the other twenty-five are called directly or indirectly when required. The translator is presently operational on an IBM 360/67 in the OS/MVT environment and uses 150K of core storage.

The translator uses four data sets. Input (NSIN) contains card images and is used to read in a nonstandard FORTRAN source program. NSLIST is a print file which is used to list the input card images and to display any of the diagnostic messages that the translator may generate. NSOUT is an output file which contains card images with conventional FORTRAN statements. The NSOUT file may be sent directly to the FORTRAN compiler. The NSWORK file is a temporary file used by the NSFORT translator.

In order to translate a nonstandard FORTRAN program into a FORTRAN program the translator (in most cases) must construct and insert several FORTRAN specification statements in the sequence of statements. To effect this the translator begins reading statements. If the statement is a SUBROUTINE, FUNCTION, or IMPLICIT statement, the translated statement is placed in the NSOUT file. After these statements, all subsequently translated statements are placed in

the NSWORK file. When an END statement is encountered the translator constructs any specification statements that are required and places them in the NSOUT file. Following this the NSWORK file is rewound and the contents are placed in the NSOUT file. The NSOUT file now contains a complete FORTRAN subprogram.

Except for the procedure described above the NSFORT translator operates on a single pass basis. Each statement in the nonstandard FORTRAN source program is read, translated and output before the next statement is translated.

As the translator processes a subprogram it stores in common storage that information relating to specification of variables, arrays, and functions which is required for subsequent statement translation. The twenty-six procedures in the translator have access to this common storage as necessary. In some cases, however, information is passed to and from procedures in argument lists of CALL statements.

The twenty-six procedures that make up the NSFORT translator are listed below in alphabetical order (except for MAIN). For each procedure the following information is given: (a) the function of the procedure; (b) the list of other procedures that call the procedure; (c) the list of other procedures called by the procedure, and (d) a brief description of the operation of the procedure.

1. MAIN (M)

   a. Function. Overall control of the translator.

   b. Calling procedures. None.

c. Procedures called. PROCCRD, ERROR.

d. Description. MAIN initializes the translator, reads card images, and terminates translation. Comment cards encountered are simply rewritten in NSOUT or NSWORK. Since the translator works with complete statements as the basic unit, MAIN also collects all continuation cards prior to calling PROCCRD to process the statement.

2. ASGMTST (A)

a. Function. Translation of assignment statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DIMENIT, DUPEIT, ERROR, EXPRSN, LITLOC, NUCARD, SEARCH.

d. Description. ASGMTST locates literals (LITLOC) and determines whether nonstandard variables are present (SEARCH). If none are present the statement is written in an output file (DUPEIT). If nonstandard variables are present, the expression to the right of the equal symbol is processed (EXPRSN). Finally a CALL statement is constructed and written on an output file (NUCARD) which will effect the assignment to the result of the expression on the right of the equal symbol to the variable on the left.

3. CALLST (C)

a. Function. Translation of CALL statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DUPEIT, EXPRSN, LITLOC, NUCARD, PARLOC, SEARCH.

d. Description. Literals are located (LITLOC) and determination is made as to the presence of nonstandard variables (SEARCH). If nonstandard variables are not present the statement is written on an output file (DUPEIT), otherwise, each argument containing nonstandard variables is processed by EXPRSN. Finally the statement is reconstructed and written on an output file (NUCARD) with the arguments being either the original arguments or the result of the processed expressions, as appropriate.

4. COMSTMT (CO)

a. Function. Translation of COMMON statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DIMENIT, DUPEIT, ERROR, NUCARD, SEARCH, SQZBLNK.

d. Description. Determination is made as to the presence of nonstandard variables (SEARCH). If nonstandard variables are not present the statement is written on an output file (DUPEIT), otherwise, the statement is translated as described in Appendix F and is written on an output file (NUCARD).

5. DATAST (DA)

a. Function. Translation of DATA statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DIMENIT, DUPEIT, ERROR, LITLOC.

d. Description. Literals are first located (LITLOC) and then the statement is examined for implicitly

declared nonstandard variables which have not been dimensioned previously. The statement is placed on an output file in its original form (DUPEIT).

6. DIMENIT (DM)

a. Function. Construction of a DIMENSION statement.

b. Calling Procedures. ASGMTST, COMSTMT, DATAST, EQUIVST, EXPRSN, NAMELST, READST.

c. Procedures called. ERROR, NUCARD.

d. Description. DIMENIT is called whenever an implicitly declared nonstandard variable is encountered. DIMENIT maintains a DIMENSION statement to which variable names are added. If the implicitly declared nonstandard variable has been encountered previously and dimensioned then control is returned to the calling procedure. If the implicitly declared nonstandard variable has not been dimensioned then it is added to the DIMENSION statement.

7. DIMENST (DI)

a. Function. Translation of DIMENSION statements.

b. Calling procedures. PROCCRD.

c. Procedures called. ERROR, NUCARD, SQZBLNK.

d. Description. DIMENST translates DIMENSION statements as described in Appendix F. The translated statement is then written on an output file (NUCARD).

8. DOSTMT (DO)

a. Function. Translation of DO statements.

b. Calling procedures. PROCCRD.

c. Procedures called. ERROR, NUCARD, SQZBLNK.

d. Description. DOSTMT replaces the end-of-range statement number with another number as described in Appendix F. The original end-of-range statement number is stored. The searching of subsequent statement numbers for the end-of-range statement is performed by PROCCRD. The insertion of a new end-of-loop CONTINUE statement is done by MAIN.

9. DUPEIT (D)

a. Function. Transfer of unaltered statements to output files.

b. Calling procedures. ASGMTST, CALLST, COMSTMT, DATAST, ENDSTMT, EQUIVST, FUNCST, IFSTMT, NAMELST, PROCCRD, READST, TYPSTMT.

c. Procedures called. None.

d. Description. When a statement from a nonstandard FORTRAN source program requires no processing or is unaltered during processing then DUPEIT is called to copy the input card images comprising the statement onto NSOUT or NSWORK as required.

10. ENDSTMT (EN)

a. Function. Completion of translation of a subprogram.

b. Calling procedures. PROCCRD.

c. Procedures called. DUPEIT, NUCARD.

d. Description. ENDSTMT is called whenever an END statement is encountered in a nonstandard FORTRAN source program. ENDSTMT constructs and writes (NUCARD) any

necessary specification statements onto NSOUT. It then concatenates the accumulated NSWORK file onto the NSOUT file. Thus NSOUT contains complete FORTRAN source subprograms. Finally ENDSTMT signals the end of a subprogram so that MAIN can reinitialize.

11. EQUIVST (EQ)

a. Function. Translation of EQUIVALENCE statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DIMENIT, DUPEIT, ERROR, NUCARD, SEARCH, SQZBLNK.

d. Description. EQUIVST translates EQUIVALENCE statements as described in Appendix F. The translated statement is written on NSWORK by DUPEIT or NUCARD.

12. ERROR (E)

a. Function. Generation of error messages on the NSLIST file.

b. Calling procedures. MAIN, ASGMTST, COMSTMT, DATAST, DIMENIT, DIMENST, DOSTMT, EQUIVST, EXPRSN, FUNCST, IFSTMT, IMPSTMT, LITLOC, NAMELST, NSTDST, PARLOC, PROCCRD, READST, TYPSTMT.

c. Procedures called. None.

d. Description. When any procedure encounters an unfamiliar construction or the translator's design limitations are exceeded, ERROR is called with an error message. This message is placed on NSLIST immediately following the statement being translated. Translation of the current statement is terminated and control is returned to MAIN.

13. EXPRSN (EX)

a. Function. Translation of arithmetic and logical expressions.

b. Calling procedures. ASGMTST, CALLST, IFSTMT.

c. Procedures called. DIMENIT, ERROR, NUCARD, PARLOC, SEARCH.

d. Description. EXPRSN is called whenever an arithmetic or logical expression containing nonstandard variables is encountered. EXPRSN translates the expression as described in Appendix F. The translation of the expression results in a single variable (usually a temporary variable constructed by the translator) which contains the result of the expression. This result is returned to the calling procedure.

14. FUNCST (F)

a. Function. Translation of explicitly declared FUNCTION statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DUPEIT, ERROR, NSTDST, NUCARD, SQZBLNK.

d. Description. Explicitly declared FUNCTION statements are translated as described in Appendix F. The translated statement is written on NSOUT by DUPEIT or NUCARD.

15. IFSTMT (IF)

a. Function. Translation of IF statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DUPEIT, ERROR, EXPRSN, LITLOC, NUCARD, PARLOC, SEARCH.

d. Description. Literals are located (LITLOC) and the arithmetic or logical conditional expression is searched for nonstandard variables (SEARCH). If necessary the expression is translated by EXPRSN. See Appendix F for a description of IF statement translation. If in a logical IF statement, the statement to be executed contains non-standard variables the control is passed back to PROCCRD and the statement is treated as the next statement.

16. IMPSTMT (I)

a. Function. Translation of IMPLICIT statements.

b. Calling procedures. PROCCRD.

c. Procedures called. ERROR, NSTDST, NUCARD, SQZBLNK.

d. Description. The NSFORT translator maintains a list of variable types corresponding to the alphabet (plus $). When initialized, variable names beginning with letters I, J, K, L. M or N will be considered INTEGER*4 variables while names beginning with any others will be considered REAL*4. When an IMPLICIT statement is encountered this alphabet list is revised as called for in the IMPLICIT statement. The translation action is performed as described in Appendix F. The translated statement is written on the NSOUT file by NUCARD.

17. LITLOC (L)

a. Function. Location and isolation of literals.

b. Calling procedures. ASGMTST, CALLST, DATAST, IFSTMT, TYPSTMT.

c. Procedures called. ERROR, SQZBLNK.

d. Description. LITLOC is called to remove literals from statements in order that the statements may be more easily processed. LITLOC locates the beginning and end of literals, stores them separately, and substitutes a code in their place. In this process all blanks except those in the literals are removed. When the translated statement is ready for output, NUCARD finds the literal codes and replaces the original literals. LITLOC will recognize either of the two ways of designating literals accepted by FORTRAN IV.

18. NAMELST (NA)

a. Function. Translation of NAMELIST statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DIMENIT, DUPEIT, ERROR, SQZBLNK.

d. Description. NAMELST processes NAMELIST statements as described in Appendix F. The original statement is written on the NSWORK file by DUPEIT.

19. NSTDST (NS)

a. Function. Processing of the NONSTANDARDi*n phrase.

b. Calling procedures. FUNCST, IMPSTMT, TYPSTMT.

c. Procedures called. ERROR.

d. Description. When the NONSTANDARDi*n phrase (or any of its optional abbreviations) is encountered NSTDST is called to determine the nonstandard type, the nonstandard size, and to check for consistency and syntax errors.

20. NUCARD (N)

a. Function. Construction of output card images.

b. Calling procedures. ASGMTST, CALLST, COMSTMT, DIMENIT, DIMENST, ENDSTMT, EQUIVST, EXPRSN, FUNCST, IFSTMT, IMPSTMT, TYPSTMT.

c. Procedures called. None.

d. Description. NUCARD first replaces all literals in the statement (see LITLOC above). NUCARD then constructs the first card image from the statement to include the statement number (if any). Continuation cards are constructed as required. The card images are then written on either the NSOUT file or the NSWORK file as appropriate.

21. PARLOC (PA)

a. Function. Location of a corresponding close parenthesis.

b. Calling procedures. CALLST, EXPRSN, IFSTMT, READST.

c. Procedures called. ERROR.

d. Description. In several circumstances it is necessary to locate a corresponding closed parenthesis; e.g. subscripts, argument lists, etc. PARLOC is called to perform this task.

22. PROCCRD (P)

a. Function. Classification of nonstandard FORTRAN statements.

b. Calling procedures. MAIN.

c. Procedures called. ASGMTST, CALLST, COMSTMT, DATAST, DIMENST, DOSTMT, DUPEIT, ENDSTMT, EQUIVST, ERROR, FUNCST, IFSTMT, IMPSTMT, NAMELST, READST, SQZBLNK, TYPSTMT.

d. Description. PROCCRD concatenates columns 7 through 72 of the card images containing a single statement. Blanks are then eliminated up to an initial delimiter. Various tests are made to determine whether the statement is an assignment statement. If so, ASGMTST is called. If not, gross classification is made based on the first two characters in the statement. In some cases, such as COMMON and COMPLEX, finer classifications must be made. If the NSFORT translator is to take no action on the particular statement then it is copied into an output file (DUPEIT). Otherwise, the appropriate procedure is called to perform the necessary translation. Additionally, PROCCRD scans statement numbers for end-of-loop statements (see DOSTMT above).

23. READST (R)

a. Function. Translation of READ statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DIMENIT, DUPEIT, ERROR, PARLOC, SEARCH, SQZBLNK.

d. Description. READST performs the action described in Appendix F. The original statement is then copied onto NSWORK by DUPEIT.

24. SEARCH (SR)

a. Function. Determination of the presence of nonstandard variable names.

b. Calling procedures. ASGMTST, CALLST, COMSTMT, EQUIVST, EXPRSN, IFSTMT, READST.

c. Procedures called. None.

d. Description. SEARCH searches statements or portions of statements for both explicitly and implicitly declared nonstandard variable names. This search is based on the lists maintained by IMPSTMT and TYPSTMT.

25. SQZBLNK (S)

a. Function. Elimination of blanks in nonstandard FORTRAN statements.

b. Calling procedures. COMSTMT, DIMENST, DOSTMT, EQUIVST, FUNCST, IMPSTMT, LITLOC, NAMELST, PROCCRD, READST.

c. Procedures called. None.

d. Description. Blanks are eliminated by compressing a statement or portion of a statement as specified by the calling procedure.

26. TYPSTMT (T)

a. Function. Translation of explicit declaration statements.

b. Calling procedures. PROCCRD.

c. Procedures called. DUPEIT, ERROR, LITLOC, NSTDST, NUCARD.

d. Description. The NSFORT translator maintains a list of all explicitly declared variables and their variable type. This list is constructed by TYPSTMT. The translation of explicit declaration statements is as described in Appendix F. The tranlated statement is written on NSWORK by NUCARD or DUPEIT as appropriate.

## APPENDIX B

### JOB CONTROL LANGUAGE USED WITH THE NSFORT TRANSLATOR

When the NSFORT translator is used on the IBM System 360/67 under OS/MVT four job steps are required. These four job steps (with the job step name) are (1) compilation of the NSFORT translator source deck (NSFORTC), (2) loading of the translator object modules (NSFORTL), (3) execution of the translator load module, or translation of the NSFORT source program (NSFORTG), and (4) compilation, loading and execution of the translated FORTRAN source program (FORTRAN).

On the following pages the sequence of job steps and the job control language is shown. Significant points to note are (1) the overlay technique in the NSFORTL step, (2) the data sets that must be defined in the NSFORTG step, (3) the passing of the FORTRAN source program from the NSFORTG step to the FORTRAN step, and (4) the condition code screening in the FORTRAN step.

By use of a user library the translator may be kept on disk on disk as a load module allowing the user to begin with the NSFORTG job step. In this case a JOBLIB DD card is required and the NSFG 10 card appropriately modified. Job control language is also shown for this use of the translator.

```
//NSFORTC EXEC PGM=IEMAA,                                          XNSFC  10
//            REGION=100K,                                         XNSFC  20
//            TIME=3                                                NSFC  30
//SYSPRINT DD  SYSOUT=A,                                           XNSFC  40
//            DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1754),              XNSFC  50
//            SPACE=(CYL,(5,1),RLSE)                                NSFC  60
//SYSUT1   DD  UNIT=(SYSDA,SEP=SYSPRINT),                          XNSFC  70
//            SPACE=(TRK,(1,30))                                    NSFC  80
//SYSLIN   DD  DSN=&LOADSET,                                       XNSFC  90
//            UNIT=(SYSDA,SEP=(SYSPRINT,SYSUT1)),                  XNSFC 100
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),                 XNSFC 110
//            SPACE=(CYL,(1,1)),                                   XNSFC 120
//            DISP=(MOD,PASS)                                       NSFC 130
//SYSIN    DD  * NSFORT TRANSLATOR SOURCE DECK FOLLOWS THIS CARD    NSFC 140

                    (NSFORT TRANSLATOR SOURCE PROGRAM DECK
                     WITH EACH PROCEDURE EXCEPT THE FIRST
                     PRECEEDED BY A CARD CONTAINING
                     "* PROCESS;" BEGINNING IN CCLUMN ONE.)

/*
//NSFORTL EXEC PGM=IEWL,                                           XNSFL  10
//            PARM='OVLY',                                         XNSFL  20
//            REGION=100K,                                         XNSFL  30
//            COND=(4,LT)                                           NSFL  40
//SYSLIB   DD  DSN=SYS1.PL1LIB,                                    XNSFL  50
//            DISP=SHR                                              NSFL  60
//SYSLMOD  DD  DSN=&GOSET(GO),                                     XNSFL  70
//            UNIT=(SYSDA,SEP=SYSLIB),                             XNSFL  80
//            SPACE=(CYL,(5,1,1),RLSE),                            XNSFL  90
//            DISP=(,PASS)                                          NSFL 100
//SYSUT1   DD  UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIB)),                  XNSFL 110
//            SPACE=(CYL,(7,1),RLSE)                                NSFL 120
//SYSPRINT DD  SYSOUT=A,                                           XNSFL 130
//            SPACE=(CYL,(1,1),RLSE)                                NSFL 140
//SYSLIN   DD  DSN=&LOADSET,                                       XNSFL 150
//            DISP=(OLD,DELETE)                                     NSFL 160
//         DD  * NSFORT TRANSLATOR OVERLAY CARDS FOLLOW THIS CARD   NSFL 170
 OVERLAY A(REGION)
 INSERT ASGMTST
 OVERLAY A
 INSERT CALLST
 OVERLAY A
 INSERT COMSTMT
 OVERLAY A
 INSERT DATAST
```

```
OVERLAY A
INSERT DIMENST
OVERLAY A
INSERT DOSTMT
OVERLAY A
INSERT ENDSTMT
OVERLAY A
INSERT EQUIVST
OVERLAY A
INSERT FUNCST
OVERLAY A
INSERT IMPSTMT
OVERLAY A
INSERT IFSTMT
OVERLAY A
INSERT NAMELST
OVERLAY A
INSERT READST
OVERLAY A
INSERT TYPSTMT
OVERLAY B(REGION)
INSERT EXPRSN
OVERLAY B
INSERT LITLOC
OVERLAY C(REGION)
INSERT DUPEIT
OVERLAY C
INSERT DIMENIT
OVERLAY C
INSERT NUCARD
OVERLAY C
INSERT NSTDST
OVERLAY C
INSERT PARLOC
OVERLAY C
INSERT SQZBLNK
OVERLAY C
INSERT SEARCH
/*
//NSFORTG EXEC PGM=*.NSFORTL.SYSLMOD,                                  XNSFGB 10
//            REGION=150K,                                             XNSFG  20
//            TIME=3,                                                  XNSFG  30
//            COND=(4,LT)                                               NSFG  40
//SYSPRINT DD  SYSOUT=A,                                               XNSFG  50
//            DCB=(RECFM=FB,LRECL=133,BLKSIZE=3325),                   XNSFG  60
//            SPACE=(CYL,(1,1),RLSE)                                    NSFG  70
//NSIN     DD  DDNAME=SYSIN                                             NSFG  80
//NSOUT    DD  DSN=&TEMPDCK,                                           XNSFG  90
```

```
//             UNIT=SYSDA,                                              XNSFG 100
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),                     XNSFG 110
//             SPACE=(CYL,(1,1),RLSE),                                  XNSFG 120
//             DISP=(NEW,PASS)                                           NSFG 130
//NSLIST   DD  SYSOUT=A,                                                XNSFG 140
//             SPACE=(CYL,(3,1),RLSE)                                    NSFG 150
//NSWORK   DD  DSN=HOLD,                                                XNSFG 160
//             UNIT=SYSDA,                                              XNSFG 170
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=80),                      XNSFG 180
//             SPACE=(CYL,(1,1),RLSE),                                  XNSFG 190
//             DISP=(NEW,DELETE)                                         NSFG 200
//SYSIN    DD  * NSFORT SOURCE DECKS FOLLOW THIS CARD                    NSFG 210

                    (NSFORT SOURCE PROGRAM DECKS)

/*
//FORTRAN EXEC FORTCLG,COND=(4,LT)                                      FTRN  10
//FORT.SYSIN DD DSN=&TEMPDCK,DISP=(OLD,DELETE)                          FTRN  20
//          DD  * FORTRAN SOURCE DECKS FOLLOW THIS CARD                 FTRN  30

                    (STANDARD FORTRAN SOURCE PROGRAM DECKS)

/*
```

```
//JOBLIB    DD  DSN=F0024.SU55,                                          XJLIB   10
//              UNIT=2314,                                               XJLIB   20
//              VOL=SER=MARY,                                            XJLIB   30
//              DISP=(OLD,PASS)                                           JLIB   40
//NSFORTG EXEC  PGM=NSFORT,                                             XNSFGA  10
//              REGION=150K,                                             XNSFG   20
//              TIME=3,                                                  XNSFG   30
//              COND=(4,LT)                                               NSFG   40
//SYSPRINT DD   SYSOUT=A,                                                XNSFG   50
//              DCB=(RECFM=FB,LRECL=133,BLKSIZE=3325),                   XNSFG   60
//              SPACE=(CYL,(1,1),RLSE)                                    NSFG   70
//NSIN      DD  DDNAME=SYSIN                                              NSFG   80
//NSOUT     DD  DSN=&TEMPDCK,                                           XNSFG   90
//              UNIT=SYSDA,                                              XNSFG  100
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),                     XNSFG  110
//              SPACE=(CYL,(1,1),RLSE),                                  XNSFG  120
//              DISP=(NEW,PASS)                                           NSFG  130
//NSLIST    DD  SYSOUT=A,                                               XNSFG  140
//              SPACE=(CYL,(3,1),RLSE)                                    NSFG  150
//NSWORK    DD  DSN=HOLD,                                               XNSFG  160
//              UNIT=SYSDA,                                              XNSFG  170
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=80),                      XNSFG  180
//              SPACE=(CYL,(1,1),RLSE),                                  XNSFG  190
//              DISP=(NEW,DELETE)                                         NSFG  200
//SYSIN     DD  * NSFORT SOURCE DECKS FOLLOW THIS CARD                    NSFG  210

                    (NSFORT SOURCE PROGRAM DECKS)

/*
//FORTRAN EXEC FORTCLG,COND=(4,LT)                                        FTRN   10
//FORT.SYSIN DD DSN=&TEMPDCK,DISP=(OLD,DELETE)                            FTRN   20
//           DD  * FORTRAN SOURCE DECKS FOLLOW THIS CARD                  FTRN   30

                    (STANDARD FORTRAN SOURCE PROGRAM DECKS)


/*
```

# APPENDIX C

## NSFORT TRANSLATOR PROGRAM LISTING

```
/* M (MAIN)  17 MAR 70  THESIS VERSION  Do Ro LITTLE  */                                                
```

```
                    || 'CDLOG(6)ALOG10(1)DLOG10(2)ARSIN(1)DARSIN(2)'      M  00490
                    || 'ARCOS(1)DARCOS(2)ATAN(1)ATAN2(1)DATAN(2)'         M  00500
                    || 'DATAN2(2)SIN(1)DSIN(2)CSIN(5)CDSIN(6)COS(1)'      M  00510
                    || 'DCOS(2)CCOS(5)CDCOS(6)TAN(1)DTAN(2)COTAN(1)'      M  00520
                    || 'DCOTAN(2)SQRT(1)DSQRT(2)CSQRT(5)CDSQRT(6)TANH(1)'M  00530
                    || 'DTANH(2)SINH(1)DSINH(2)COSH(1)DCOSH(2)ERF(1)'     M  00540
                    || 'DERF(2)ERFC(1)DERFC(2)GAMMA(1)DGAMMA(2)ALGAMA(1)'M  00550
                    || 'DLGAMA(2)MOD(4)AMOD(1)DMOD(2)IABS(4)ABS(1)'       M  00560
                    || 'DABS(2)CABS(1)CDABS(2)INT(4)AINT(1)IDINT(4)'      M  00570
                    || 'AMAX0(1)AMAX1(1)MAX0(4)MAX1(4)DMAX1(2)AMIN0(1)'   M  00580
                    || 'AMIN1(1)MIN0(4)MIN1(4)DMIN1(2)FLOAT(1)DFLOAT(2)'  M  00590
                    || 'IFIX(4)HFIX(3)SIGN(1)ISIGN(4)DSIGN(2)DIM(1)'      M  00600
                    || 'IDIM(4)SNGL(1)REAL(1)AIMAG(1)DBLE(2)CMPLX(5)'     M  00610
                    || 'DCMPLX(6)CONJG(5)DCONJG(6)';                      M  00620
          CONTNO = 99999;                                                 M  00630
          ON ENDFILE (NSIN) BEGIN;                                        M  00640
            IF CC1¬='C' THEN DO;                                          M  00650
              CALL PROCCRD;                                               M  00660
              IF LOOPEND='1'B THEN CALL LUPEND;                           M  00670
              END;                                                        M  00680
            GO TO M40;                                                    M  00690
            END;                                                          M  00700
          OPEN FILE (NSWORK) OUTPUT;                                      M  00710
          PUT FILE (NSLIST) PAGE EDIT ('NONSTANDARD FORTRAN SOURCE ',     M  00720
              'PROGRAM LISTING WITH ERRORS NOTED:') (A);                  M  00730
          PUT FILE (NSLIST) SKIP(3) EDIT (' ') (A);                       M  00740
          PUT FILE (NSOUT) EDIT ('C', (79)' ', 'C', (79)' ',              M  00750
              'C          GENERATED FORTRAN SOURCE PROGRAM', (39)' ', 'C',M  00760
              (79)' ', 'C', (79)' ') (A);                                 M  00770
M10:      CARDNO = 1;                                                     M  00780
          GET FILE (NSIN) EDIT (CARD(1)) (A(80));                         M  00790
          CC1 = SUBSTR(CARD(1), 1, 1);                                    M  00800
          CC6 = SUBSTR(CARD(1), 6, 1);                                    M  00810
M20:      IF CC1='C' THEN DO;                                             M  00820
            PUT FILE (NSLIST) SKIP EDIT (CARD(1)) (X(20), A(80));         M  00830
            IF ACTFILE='OUT '                                             M  00840
              THEN PUT FILE (NSOUT) EDIT (CARD(1)) (A(80));               M  00850
              ELSE PUT FILE (NSWORK) EDIT (CARD(1)) (A(80));              M  00860
            GO TO M10;                                                    M  00870
            END;                                                          M  00880
          IF CC6¬=' ' & CC6¬='0' THEN DO;                                 M  00890
            CALL ERROR ('M -01: UNEXPECTED CONTINUATION CARD.');          M  00900
            GO TO M10;                                                    M  00910
            END;                                                          M  00920
M30:      PUT FILE (NSLIST) SKIP EDIT (CARD(CARDNO)) (X(20), A(80));      M  00930
          GET FILE (NSIN) EDIT (NEXTCARD) (A(80));                        M  00940
          CC1 = SUBSTR(NEXTCARD, 1, 1);                                   M  00950
          CC6 = SUBSTR(NEXTCARD, 6, 1);                                   M  00960
```

```
          IF CC1='C' | CC6=' ' | CC6='0' THEN DO;                                  M  00970
            CALL PROCCRD;                                                          M  00980
            IF LOOPEND='1'B THEN CALL LUPEND;                                      M  00990
            IF PROGEND='1'B THEN CALL INIT;                                        M  01000
            CARDNO = 1;                                                            M  01010
            CARD(1) = NEXTCARD;                                                    M  01020
            GO TO M20;                                                             M  01030
            END;                                                                   M  01040
          IF CARDNO = 20 THEN DO;                                                  M  01050
            CARD(21) = NEXTCARD;                                                   M  01060
            CALL ERROR ('M -02: EXCESSIVE CONTINUATION CARDS OVER 19.');           M  01070
            GO TO M10;                                                             M  01080
            END;                                                                   M  01090
          CARDNO = CARDNO + 1;                                                     M  01100
          CARD(CARDNO) = NEXTCARD;                                                 M  01110
          GO TO M30;                                                               M  01120
INIT:     PROC;                                                                    M  01130
          CALSTMT = '0'B;                                                          M  01140
          ERRFLG = '0'B;                                                           M  01150
          LOOPEND = '0'B;                                                          M  01160
          OLDSTMT = '0'B;                                                          M  01170
          PROGEND = '0'B;                                                          M  01180
          DIMLIST = '|';                                                           M  01190
          DIMSTMT = (0)' ';                                                        M  01200
          EXPLIST = SUBSTR(EXPLIST, 1, 650);                                       M  01210
          EXPSTMT = (0)' ';                                                        M  01220
          IMPCODE = '111111114444441111111111111';                                 M  01230
          IMPNSL = (0)' ';                                                         M  01240
          LOOPNOS = (0)' ';                                                        M  01250
          NSFNS = '|';                                                             M  01260
          EXPNSL = '|';                                                            M  01270
          CONTNO = 99999;                                                          M  01280
          DO I=1 TO 6;                                                             M  01290
            NSDIM(I) = '0';                                                        M  01300
            NSTN(I, 1) = 0;                                                        M  01310
            NSTN(I, 2) = 0;                                                        M  01320
            TYPN(I, 1) = 0;                                                        M  01330
            TYPN(I, 2) = 0;                                                        M  01340
            END;                                                                   M  01350
          DO I=7 TO 8;                                                             M  01360
            TYPN(I, 1) = 0;                                                        M  01370
            TYPN(I, 2) = 0;                                                        M  01380
            END;                                                                   M  01390
          PUT FILE (NSLIST) PAGE EDIT ('NONSTANDARD FORTRAN SOURCE ',              M  01400
               'PROGRAM LISTING WITH ERRORS NOTED:') (A);                        M  01410
          PUT FILE (NSLIST) SKIP(3) EDIT (' ') (A);                                M  01420
          PUT FILE (NSOUT) EDIT ('C', (79)' ', 'C',                                M  01430
               (79)' ', 'C        GENERATED FORTRAN SOURCE PROGRAM',               M  01440
```

```
                (39)' ', 'C', (79)' ', 'C', (79)' ') (A);                    M  01450
          RETURN;                                                            M  01460
          END INIT;                                                          M  01470
LUPEND:   PROC;                                                              M  01480
          DOT = INDEX(LOOPNOS, '.');                                         M  01490
          STMTNO = SUBSTR(LOOPNOS, DOT+1, 5);                                M  01500
          IF LENGTH(LOOPNOS)<12                                              M  01510
            THEN LOOPNOS = (6)' ';                                           M  01520
            ELSE LOOPNOS = SUBSTR(LOOPNOS, DOT+6);                           M  01530
          STMT = 'CONTINUE';                                                 M  01540
          CALL NUCARD;                                                       M  01550
          LOOPEND = '0'B;                                                    M  01560
          RETURN;                                                            M  01570
          END LUPEND;                                                        M  01580
M40:      END MAIN;                                                          M  01590
```

```
/*  A  (ASGMTST)  17 MAR 70  THESIS VERSION  D. R. LITTLE  */                    A  00010
ASGMTST:PROC;                                                                    A  00020
        DCL  ALPHBET      CHAR(27)          EXT,                               A  00030
             DIMLIST      CHAR(1000)VAR     EXT,                               A  00040
             EXPLIST      CHAR(2000)VAR     EXT,                               A  00050
             EXPR         CHAR(1320)VAR     EXT,                               A  00060
             IMPCODE      CHAR(27)          EXT,                               A  00070
             NSDIM(6)     CHAR(3)VAR        EXT,                               A  00080
             RSLT         CHAR(30)VAR       EXT,                               A  00090
             RSLTYP       CHAR(1)           EXT,                               A  00100
             STMT         CHAR(1320)VAR     EXT,                               A  00110
             OPID         CHAR(6)VAR,                                          A  00120
             OPERAND      CHAR(30)VAR,                                         A  00130
             OPRNDTYP     CHAR(1),                                             A  00140
             ERRFLG       BIT(1)            EXT,                               A  00150
             NSSTAT       BIT(1)            EXT,                               A  00160
             OLDSTMT      BIT(1)            EXT,                               A  00170
             EQPOS        FIXED,                                               A  00180
             EXPCHK       FIXED,                                               A  00190
             NSTNO        FIXED,                                               A  00200
             NXTOPPR      FIXED,                                               A  00210
             OPIDLN       FIXED;                                               A  00220
        IF OLDSTMT='1'B THEN DO;                                                 A  00230
          OLDSTMT = '0'B;                                                        A  00240
          GO TO A10;                                                             A  00250
          END;                                                                   A  00260
        CALL LITLOC;                                                             A  00270
        IF ERRFLG='1'B THEN GO TO A20;                                           A  00280
        CALL SEARCH (STMT);                                                      A  00290
        IF NSSTAT='0'B THEN DO;                                                  A  00300
          CALL DUPEIT;                                                           A  00310
          RETURN;                                                                A  00320
          END;                                                                   A  00330
A10:    EQPOS = INDEX(STMT, '=');                                                A  00340
        NXTOPPR = INDEX(STMT, '(');                                              A  00350
        IF NXTOPPR=0 THEN NXTOPPR = 1321;                                        A  00360
        IF EQPOS>31 THEN DO;                                                     A  00370
          CALL ERROR ('A -01: DESIGN PARAMETER EXCEEDED: OPERAND.');             A  00380
          RETURN;                                                                A  00390
          END;                                                                   A  00400
        OPERAND = SUBSTR(STMT, 1, EQPOS-1);                                      A  00410
        IF NXTOPPR<EQPOS                                                         A  00420
          THEN DO;                                                               A  00430
            OPID = SUBSTR(STMT, 1, NXTOPPR-1);                                   A  00440
            IF INDEX(DIMLIST, '|'||OPID||'|')=0 THEN DO;                         A  00450
              CALL ERROR ('A -02: STMT FN CONTAINS NONSTD.');                    A  00460
              RETURN;                                                            A  00470
              END;                                                               A  00480
```

```
        END;                                                                A  00490
      ELSE OPID = OPERAND;                                                  A  00500
    EXPR = SUBSTR(STMT, EQPOS+1);                                           A  00510
    CALL EXPRSN;                                                            A  00520
    IF ERRFLG='1'B THEN GO TO A20;                                          A  00530
    OPIDLN = LENGTH(OPID);                                                  A  00540
    EXPCHK = INDEX(EXPLIST, ')'||OPID||'(');                                A  00550
    IF EXPCHK>0                                                             A  00560
      THEN OPRNDTYP = SUBSTR(EXPLIST, EXPCHK+OPIDLN+2, 1);                  A  00570
      ELSE DO;                                                              A  00580
        OPRNDTYP = SUBSTR(IMPCODE, INDEX(ALPHBET,                           A  00590
                   SUBSTR(OPID, 1, 1)), 1);                                 A  00600
        NSTNO = INDEX(ALPHBET, OPRNDTYP);                                   A  00610
        IF NSTNO>0 THEN CALL DIMENIT (OPID, NSTNO);                         A  00620
        IF ERRFLG='1'B THEN GO TO A20;                                      A  00630
        END;                                                                A  00640
    IF (INDEX('78', OPRNDTYP)>0 & INDEX('78', RSLTYP)=0) | (INDEX(          A  00650
        '78', OPRNDTYP)=0 & INDEX('78', RSLTYP)>0) THEN DO;                 A  00660
      CALL ERROR ('A -03: ASGMT STMT HAS LOG VAR ON ONLY 1 SIDE.');         A  00670
      RETURN;                                                               A  00680
      END;                                                                  A  00690
    IF INDEX(ALPHBET, OPRNDTYP)>0 THEN IF NXTOPPR<EQPOS                     A  00700
      THEN DO;                                                              A  00710
        IF LENGTH(OPERAND)>28 THEN DO;                                      A  00720
          CALL ERROR('A -04: DESIGN PARAMETER EXCEEDED: OPERAND.');         A  00730
          RETURN;                                                           A  00740
          END;                                                              A  00750
        OPERAND = SUBSTR(OPERAND, 1, NXTOPPR) || '1,' ||                    A  00760
                  SUBSTR(OPERAND, NXTOPPR+1);                               A  00770
        END;                                                                A  00780
      ELSE OPERAND = OPERAND || '(1)';                                      A  00790
    IF INDEX('78', OPRNDTYP)>0                                              A  00800
      THEN STMT = OPERAND || ' = ' || RSLT;                                 A  00810
      ELSE STMT = 'CALL N$' || OPRNDTYP || RSLTYP || '   (' ||             A  00820
                  OPERAND || ', ' || RSLT || ')';                           A  00830
    CALL NUCARD;                                                            A  00840
    RETURN;                                                                 A  00850
A20: ERRFLG = '0'B;                                                         A  00860
    RETURN;                                                                 A  00870
    END ASGMTST;                                                            A  00880
```

```
/*  C  (CALLST)   26 JAN 70  THESIS VERSION  Do Ro LITTLE  */                    C  00010
CALLST: PROC;                                                                   C  00020
        DCL EXPR          CHAR(1320)VAR  EXT,                                   C  00030
            RSLT          CHAR(30)VAR    EXT,                                   C  00040
            STMT          CHAR(1320)VAR  EXT,                                   C  00050
            NEWARGS       CHAR(1314)VAR,                                        C  00060
            OLDARGS       CHAR(1314)VAR,                                        C  00070
            SUBSTMT       CHAR(13)VAR,                                          C  00080
            SYMB          CHAR(1),                                              C  00090
            CALSTMT       BIT(1)         EXT,                                   C  00100
            ERRFLG        BIT(1)         EXT,                                   C  00110
            NSSTAT        BIT(1)         EXT,                                   C  00120
            OLDSTMT       BIT(1)         EXT,                                   C  00130
            NSTN(6,2)     FIXED          EXT,                                   C  00140
            TYPN(8,2)     FIXED          EXT,                                   C  00150
            ENDPR         FIXED,                                                C  00160
            FSTPR         FIXED,                                                C  00170
            NXTDLM        FIXED,                                                C  00180
            STRT          FIXED,                                                C  00190
            SYMLOC        FIXED;                                                C  00200
        FSTPR = INDEX(STMT, '(');                                               C  00210
        IF FSTPR=0 THEN GO TO C50;                                              C  00220
        NEWARGS = (0)' ';                                                       C  00230
        IF OLDSTMT='1'B THEN DO;                                                C  00240
          OLDSTMT = '0'B;                                                       C  00250
          GO TO C10;                                                            C  00260
          END;                                                                  C  00270
        CALL LITLOC;                                                            C  00280
        IF ERRFLG='1'B THEN GO TO C60;                                          C  00290
C10:    OLDARGS = SUBSTR(STMT, FSTPR+1);                                        C  00300
        OLDARGS = SUBSTR(OLDARGS, 1, LENGTH(OLDARGS)-1) || ';';                 C  00310
        CALL SEARCH (OLDARGS);                                                  C  00320
        IF NSSTAT='0'B THEN GO TO C50;                                          C  00330
        SUBSTMT = 'CALL ' || SUBSTR(STMT, 5, FSTPR-5) || ' (';                  C  00340
        CALSTMT = '1'B;                                                         C  00350
C20:    STRT = 1;                                                               C  00360
C30:    NXTDLM = 1321;                                                          C  00370
        DO SYMB='(',',',';';                                                    C  00380
          SYMLOC = INDEX(SUBSTR(OLDARGS, STRT), SYMB);                          C  00390
          IF SYMLOC=0                                                           C  00400
            THEN SYMLOC = 1321;                                                 C  00410
            ELSE SYMLOC = SYMLOC + STRT - 1;                                    C  00420
          IF  SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                               C  00430
          END;                                                                  C  00440
        SYMB = SUBSTR(OLDARGS, NXTDLM, 1);                                      C  00450
        IF SYMB='(' THEN DO;                                                    C  00460
          CALL PARLOC (OLDARGS, NXTDLM, ENDPR);                                 C  00470
          IF ERRFLG='1'B THEN GO TO C60;                                        C  00480
```

```
            STRT = ENDPR + 1;                                                    C  00490
            GO TO C30;                                                           C  00500
            END;                                                                 C  00510
          EXPR = SUBSTR(OLDARGS, 1, NXTDLM-1);                                   C  00520
          IF SYMB = ','                                                          C  00530
            THEN OLDARGS = SUBSTR(OLDARGS, NXTDLM+1);                            C  00540
            ELSE CALSTMT = '0'B;                                                 C  00550
          CALL SEARCH (EXPR);                                                    C  00560
          IF NSSTAT='0'B THEN DO;                                                C  00570
            NEWARGS = NEWARGS || ', ' || EXPR;                                   C  00580
            IF SYMB =';' THEN DO;                                                C  00590
              DO I=1 TO 8;                                                       C  00600
                IF I<7 THEN DO;                                                  C  00610
                  IF NSTN(I,1)>NSTN(I,2) THEN NSTN(I,2) = NSTN(I,1);             C  00620
                  NSTN(I,1) = 0;                                                 C  00630
                  END;                                                           C  00640
                IF TYPN(I,1)>TYPN(I,2) THEN TYPN(I,2) = TYPN(I,1);               C  00650
                TYPN(I,1) = 0;                                                   C  00660
                END;                                                             C  00670
              GO TO C40;                                                         C  00680
              END;                                                               C  00690
            GO TO C20;                                                           C  00700
            END;                                                                 C  00710
          CALL EXPRSN;                                                           C  00720
          IF ERRFLG='1'B THEN GO TO C60;                                         C  00730
          NEWARGS = NEWARGS || ', ' || RSLT;                                     C  00740
          IF SYMB=',' THEN GO TO C20;                                            C  00750
C40:      STMT = SUBSTMT || SUBSTR(NEWARGS, 3) || ')';                           C  00760
          CALL NUCARD;                                                           C  00770
          RETURN;                                                                C  00780
C50:      CALL DUPEIT;                                                           C  00790
          RETURN;                                                                C  00800
C60:      ERRFLG = '0'B;                                                         C  00810
          RETURN;                                                                C  00820
          END CALLST;                                                            C  00830
```

```
/*  CO (COMSTMT)  17 MAR 70  THESIS VERSION  D. R. LITTLE  */                    CO 00010
COMSTMT:PROC;                                                                    CO 00020
        DCL ALPHBET       CHAR(27)          EXT,                               CO 00030
            EXPLIST       CHAR(2000)VAR     EXT,                               CO 00040
            IMPCODE       CHAR(27)          EXT,                               CO 00050
            IMPNSL        CHAR(27)VAR       EXT,                               CO 00060
            NSDIM(6)      CHAR(3)VAR        EXT,                               CO 00070
            STMT          CHAR(1320)VAR     EXT,                               CO 00080
            CHAR1         CHAR(1),                                             CO 00090
            SYMB          CHAR(1),                                             CO 00100
            VARID         CHAR(6)VAR,                                          CO 00110
            VARTYP        CHAR(1),                                             CO 00120
            NSSTAT        BIT(1)            EXT,                               CO 00130
            EXPCHK        FIXED,                                               CO 00140
            NSTNO         FIXED,                                               CO 00150
            NXTDLM        FIXED,                                               CO 00160
            STRT          FIXED,                                               CO 00170
            SYMLOC        FIXED,                                               CO 00180
            SEARCH        ENTRY(CHAR(1320)VAR),                                CO 00190
            SQZBLNK       ENTRY(FIXED,FIXED);                                  CO 00200
        CALL SQZBLNK (1, 1321);                                                  CO 00210
        CALL SEARCH (SUBSTR(STMT, 7));                                           CO 00220
        IF NSSTAT='0'B THEN DO;                                                  CO 00230
          CALL DUPEIT;                                                           CO 00240
          RETURN;                                                                CO 00250
          END;                                                                   CO 00260
        STMT = STMT || ';';                                                      CO 00270
        NXTDLM = 7;                                                              CO 00280
        STRT = 7;                                                                CO 00290
        IF SUBSTR(STMT, 7, 1)='/' THEN DO;                                       CO 00300
C10:      STRT = INDEX(SUBSTR(STMT, NXTDLM+1), '/') + NXTDLM + 1;                CO 00310
          IF STRT=NXTDLM+1 THEN DO;                                              CO 00320
            CALL ERROR ('CO-01: NO CLOSE SLASH FOUND.');                         CO 00330
            RETURN;                                                              CO 00340
            END;                                                                 CO 00350
          END;                                                                   CO 00360
C20:    IF INDEX(ALPHBET, SUBSTR(STMT, STRT, 1))=0 THEN DO;                      CO 00370
          CALL ERROR ('CO-02: ILLEGAL CHAR FOLLOWS COMMA OR SLASH.');            CO 00380
          RETURN;                                                                CO 00390
          END;                                                                   CO 00400
        NXTDLM = 1321;                                                           CO 00410
        DO SYMB='(',',','/',';';                                                 CO 00420
          SYMLOC = INDEX(SUBSTR(STMT, STRT), SYMB);                              CO 00430
          IF SYMLOC=0                                                            CO 00440
            THEN SYMLOC = 1321;                                                  CO 00450
            ELSE SYMLOC = SYMLOC + STRT - 1;                                     CO 00460
          IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                                 CO 00470
          END;                                                                   CO 00480
```

```
            IF NXTDLM-STRT>6 THEN DO;                                          CO 00490
              CALL ERROR ('CO-03: VAR IDENTIFIER HAS > 6 CHAR.');              CO 00500
              RETURN;                                                          CO 00510
              END;                                                             CO 00520
            SYMB = SUBSTR(STMT, NXTDLM, 1);                                    CO 00530
            VARID = SUBSTR(STMT, STRT, NXTDLM-STRT);                           CO 00540
            EXPCHK = INDEX(EXPLIST, ')'||VARID||'(');                          CO 00550
            IF EXPCHK>0 THEN DO;                                               CO 00560
              VARTYP = SUBSTR(EXPLIST, EXPCHK+NXTDLM-STRT+2, 1);               CO 00570
              NSTNO = INDEX(ALPHBET, VARTYP);                                  CO 00580
              GO TO C30;                                                       CO 00590
              END;                                                             CO 00600
            CHAR1 = SUBSTR(VARID, 1, 1);                                       CO 00610
            IF INDEX(IMPNSL, CHAR1)>0                                          CO 00620
              THEN DO;                                                         CO 00630
                NSTNO = INDEX(ALPHBET, SUBSTR(IMPCODE, INDEX(ALPHBET,          CO 00640
                        CHAR1), 1));                                           CO 00650
                CALL DIMENIT (VARID, NSTNO);                                   CO 00660
                END;                                                           CO 00670
              ELSE NSTNO = 0;                                                  CO 00680
C30:        IF SYMB='(' THEN DO;                                               CO 00690
              IF NSTNO>0 THEN STMT = SUBSTR(STMT, 1, NXTDLM) ||                CO 00700
                   NSDIM(NSTNO) || ',' || SUBSTR(STMT, NXTDLM+1);              CO 00710
              STRT = INDEX(SUBSTR(STMT, NXTDLM), ')') + NXTDLM;                CO 00720
              IF STRT=NXTDLM THEN DO;                                          CO 00730
                CALL ERROR ('CO-04: NO CLOSE PAREN FOUND.');                   CO 00740
                RETURN;                                                        CO 00750
                END;                                                           CO 00760
              NXTDLM = STRT;                                                   CO 00770
              SYMB = SUBSTR(STMT, NXTDLM, 1);                                  CO 00780
              END;                                                             CO 00790
            IF SYMB=',' THEN DO;                                               CO 00800
              STRT = NXTDLM + 1;                                               CO 00810
              GO TO C20;                                                       CO 00820
              END;                                                             CO 00830
            IF SYMB='/' THEN GO TO C10;                                        CO 00840
            IF SYMB=';' THEN DO;                                               CO 00850
              STMT = SUBSTR(STMT, 1, NXTDLM-1);                                CO 00860
              CALL NUCARD;                                                     CO 00870
              RETURN;                                                          CO 00880
              END;                                                             CO 00890
            CALL ERROR ('CO-05: ILLEGAL CHAR FOLLOWS CLOSE PAREN.');           CO 00900
            RETURN;                                                            CO 00910
            END CCMSTMT;                                                       CO 00920
```

```
/*  D  (DUPEIT)   26 JAN 70  THESIS VERSION  Do Ro LITTLE  */          D  00010
DUPEIT: PROC;                                                          D  00020
        DCL  ACTFILE      CHAR(4)          EXT,                      D  00030
             CARD(21)     CHAR(80)         EXT,                      D  00040
             CARDNO       FIXED            EXT,                      D  00050
             NSOUT        FILE OUTPUT      EXT,                      D  00060
             NSWORK       FILE             EXT;                      D  00070
        DO I=1 TO CARDNO;                                              D  00080
          IF ACTFILE='OUT '                                            D  00090
            THEN PUT FILE (NSOUT) EDIT (CARD(I)) (A(80));              D  00100
            ELSE PUT FILE (NSWORK) EDIT (CARD(I)) (A(80));             D  00110
          END;                                                         D  00120
        RETURN;                                                        D  00130
        END DUPEIT;                                                    D  00140
```

```
/*  DA (DATAST)   26 JAN 70  THESIS VERSION  D. R. LITTLE  */                DA 00010
DATAST: PROC;                                                               DA 00020
        DCL ALPHBET      CHAR(27)          EXT,                             DA 00030
            EXPLIST      CHAR(2000)VAR     EXT,                             DA 00040
            IMPCODE      CHAR(27)          EXT,                             DA 00050
            IMPNSL       CHAR(27)          EXT,                             DA 00060
            STMT         CHAR(1320)VAR     EXT,                             DA 00070
            CHAR1        CHAR(1),                                           DA 00080
            SYMB         CHAR(1),                                           DA 00090
            VARID        CHAR(6)VAR,                                        DA 00100
            ERRFLG       BIT(1)            EXT,                             DA 00110
            OLDSTMT      BIT(1)            EXT,                             DA 00120
            NSTNO        FIXED,                                             DA 00130
            STRT         FIXED,                                             DA 00140
            SYMLOC       FIXED;                                             DA 00150
        IF IMPNSL=(0)' ' THEN GO TO D40;                                    DA 00160
        IF OLDSTMT='1'B THEN GO TO D10;                                     DA 00170
        CALL LITLOC;                                                        DA 00180
        IF ERRFLG='1'B THEN GO TO D50;                                      DA 00190
D10:    STMT = STMT || ';';                                                 DA 00200
        STRT = 5;                                                           DA 00210
D20:    NXTDLM = 1321;                                                      DA 00220
        DO SYMB='(',',','/',';';                                            DA 00230
          SYMLOC = INDEX(SUBSTR(STMT, STRT), SYMB);                         DA 00240
          IF SYMLOC=0                                                       DA 00250
            THEN SYMLOC = 1321;                                             DA 00260
            ELSE SYMLOC = SYMLOC + STRT - 1;                                DA 00270
          IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                            DA 00280
          END;                                                              DA 00290
        IF NXTDLM-STRT>6 THEN DO;                                           DA 00300
          CALL ERROR ('DA-01: VAR IDENTIFIER HAS > 6 CHAR.');               DA 00310
          RETURN;                                                           DA 00320
          END;                                                              DA 00330
        VARID = SUBSTR(STMT, STRT, NXTDLM-STRT);                            DA 00340
        IF INDEX(EXPLIST, ')'||VARID||'(')>0 THEN GO TO D30;                DA 00350
        CHAR1 = SUBSTR(STMT, STRT, 1);                                      DA 00360
        IF INDEX(IMPNSL, CHAR1)>0 THEN DO;                                  DA 00370
          NSTNO = INDEX(ALPHBET, SUBSTR(IMPCODE, INDEX(ALPHBET, CHAR1),     DA 00380
                  1));                                                      DA 00390
          CALL DIMENIT (VARID, NSTNO);                                      DA 00400
          IF ERRFLG='1'B THEN GO TO D50;                                    DA 00410
          END;                                                              DA 00420
D30:    SYMB = SUBSTR(STMT, NXTDLM, 1);                                     DA 00430
        IF SYMB='(' THEN DO;                                                DA 00440
          STRT = INDEX(SUBSTR(STMT, NXTDLM), ')') + NXTDLM;                 DA 00450
          IF STRT=NXTDLM THEN DO;                                           DA 00460
            CALL ERROR ('DA-02: NO CLOSE PAREN FOUND.');                    DA 00470
            RETURN;                                                         DA 00480
```

```
          END;                                                                    DA 00490
        NXTDLM = STRT;                                                            DA 00500
        SYMB = SUBSTR(STMT, NXTDLM,  1);                                          DA 00510
        END;                                                                      DA 00520
      IF SYMB='/' THEN DO;                                                        DA 00530
        STRT = INDEX(SUBSTR(STMT, NXTDLM+1), '/') + NXTDLM + 1;                   DA 00540
        IF STRT=NXTDLM+1 THEN DO;                                                 DA 00550
          CALL ERROR ('DA-03: NO CLOSE SLASH FOUND.');                            DA 00560
          RETURN;                                                                 DA 00570
          END;                                                                    DA 00580
        IF SUBSTR(STMT, STRT, 1)=';' THEN DO;                                     DA 00590
D40:      IF OLDSTMT='1'B                                                         DA 00600
            THEN OLDSTMT = '0'B;                                                  DA 00610
            ELSE CALL DUPEIT;                                                     DA 00620
          RETURN;                                                                 DA 00630
          END;                                                                    DA 00640
        NXTDLM = STRT;                                                            DA 00650
        SYMB = SUBSTR(STMT, NXTDLM, 1);                                           DA 00660
        END;                                                                      DA 00670
      IF SYMB =',' THEN DO;                                                       DA 00680
        STRT = NXTDLM + 1;                                                        DA 00690
        IF INDEX(ALPHBET, SUBSTR(STMT, STRT, 1))>0 THEN GO TO D20;                DA 00700
        END;                                                                      DA 00710
      CALL ERRCR ('DA-04: DATA STMT SYNTAX ERROR.');                              DA 00720
      RETURN;                                                                     DA 00730
D50:  ERRFLG = '0'B;                                                              DA 00740
      RETURN;                                                                     DA 00750
      END DATAST;                                                                 DA 00760
```

```
/*  DI (DIMENST)  20 MAR 70  THESIS VERSION  D. R. LITTLE  */                DI 00010
DIMENST:PROC;                                                               DI 00020
        DCL  ALPHBET        CHAR(27)         EXT,                           DI 00030
             DIMLIST        CHAR(1000)VAR    EXT,                           DI 00040
             EXPLIST        CHAR(2000)VAR    EXT,                           DI 00050
             IMPCODE        CHAR(27)         EXT,                           DI 00060
             IMPNSL         CHAR(27)VAR      EXT,                           DI 00070
             NSDIM(6)       CHAR(3)VAR       EXT,                           DI 00080
             STMT           CHAR(1320)VAR    EXT,                           DI 00090
             VARID          CHAR(6)VAR,                                     DI 00100
             VARTYP         CHAR(1),                                        DI 00110
             CLPR           FIXED,                                          DI 00120
             EXPCHK         FIXED,                                          DI 00130
             OPPR           FIXED,                                          DI 00140
             STRT           FIXED,                                          DI 00150
             SQZBLNK        ENTRY(FIXED,FIXED);                             DI 00160
        CALL SQZBLNK (1, 1321);                                             DI 00170
        STRT = 10;                                                          DI 00180
D10:    OPPR = INDEX(SUBSTR(STMT, STRT), '(') + STRT - 1;                   DI 00190
        IF OPPR=STRT-1 THEN DO;                                             DI 00200
          CALL ERROR ('DI-01: OPEN PAREN NOT FOUND WHEN EXPECTED.');        DI 00210
          RETURN;                                                           DI 00220
          END;                                                              DI 00230
        VARID = SUBSTR(STMT, STRT, OPPR-STRT);                              DI 00240
        EXPCHK = INDEX(EXPLIST, ')'||VARID||'(');                           DI 00250
        IF EXPCHK¬=0 THEN DO;                                               DI 00260
          VARTYP = SUBSTR(EXPLIST, EXPCHK+OPPR-STRT+2, 1);                  DI 00270
          IF INDEX(ALPHBET, VARTYP)>0 THEN DO;                              DI 00280
            CALL ERROR('DI-02: EXP DCL NONSTD VAR IN DIMEN STMT.');         DI 00290
            RETURN;                                                         DI 00300
            END;                                                            DI 00310
          GO TO D20;                                                        DI 00320
          END;                                                              DI 00330
        IF INDEX(IMPNSL, SUBSTR(VARID, 1, 1))>0 THEN DO;                    DI 00340
          VARTYP = SUBSTR(IMPCODE, INDEX(ALPHBET, SUBSTR(VARID, 1, 1)),     DI 00350
                   1);                                                      DI 00360
          STMT = SUBSTR(STMT, 1, OPPR) || NSDIM(INDEX(ALPHBET, VARTYP))     DI 00370
                 || ',' || SUBSTR(STMT, OPPR+1);                            DI 00380
          GO TO D20;                                                        DI 00390
          END;                                                              DI 00400
D20:    IF LENGTH(DIMLIST)>993 THEN DO;                                     DI 00410
          CALL ERROR ('DI-03: DESIGN PARAMETERS EXCEEDED: DIMLIST.');       DI 00420
          RETURN;                                                           DI 00430
          END;                                                              DI 00440
        DIMLIST = DIMLIST || VARID || '|';                                  DI 00450
        CLPR = INDEX(SUBSTR(STMT, STRT), ')') + STRT - 1;                   DI 00460
        IF CLPR=STRT-1 THEN DO;                                             DI 00470
          CALL ERROR ('DI-04: NO CLOSE PAREN FOUND WHEN EXPECTED.');        DI 00480
```

```
   RETURN;                                                                  DI 00490
   END;                                                                     DI 00500
IF LENGTH(STMT)=CLPR THEN DO;                                               DI 00510
   STMT = SUBSTR(STMT, 1, 9) || ' ' || SUBSTR(STMT, 10);                    DI 00520
   CALL NUCARD;                                                             DI 00530
   RETURN;                                                                  DI 00540
   END;                                                                     DI 00550
STRT = CLPR + 2;                                                            DI 00560
GO TO D10;                                                                  DI 00570
END DIMENST;                                                                DI 00580
```

```
/*  DM (DIMENIT)  26 JAN 70  THESIS VERSION  D. R. LITTLE  */                DM 00010
DIMENIT:PROC (VARID, NSTNO);                                                 DM 00020
        DCL ACTFILE       CHAR(4)          EXT,                            DM 00030
            DIMLIST       CHAR(1000)VAR    EXT,                            DM 00040
            DIMSTMT       CHAR(1320)VAR    EXT,                            DM 00050
            NSDIM(6)      CHAR(3)VAR       EXT,                            DM 00060
            STMT          CHAR(1320)VAR    EXT,                            DM 00070
            STMTHLD       CHAR(1320)VAR,                                   DM 00080
            VARID         CHAR(6)VAR,                                      DM 00090
            ERRFLG        BIT(1)           EXT,                            DM 00100
            NSTNO         FIXED;                                           DM 00110
        IF INDEX(DIMLIST, '|'||VARID||'|')>0 THEN RETURN;                    DM 00120
        IF LENGTH(DIMSTMT)>1308 THEN DO;                                     DM 00130
          STMTHLD = STMT;                                                    DM 00140
          STMT = DIMSTMT;                                                    DM 00150
          ACTFILE = 'OUT ';                                                  DM 00160
          CALL NUCARD;                                                       DM 00170
          ACTFILE = 'WORK';                                                  DM 00180
          STMT = STMTHLD;                                                    DM 00190
          DIMSTMT = (0)' ';                                                  DM 00200
          END;                                                               DM 00210
        IF DIMSTMT=(0)' '                                                    DM 00220
          THEN DIMSTMT = 'DIMENSION ' || VARID || '(' || NSDIM(NSTNO)        DM 00230
                         || ')';                                             DM 00240
          ELSE DIMSTMT = DIMSTMT || ',' || VARID || '(' ||                   DM 00250
                         NSDIM(NSTNO) || ')';                                DM 00260
        IF LENGTH(DIMLIST)>993 THEN DO;                                      DM 00270
          CALL ERROR ('DM-01: DESIGN PARAMETER EXCEEDED: DIMLIST.');         DM 00280
          ERRFLG = '1'B;                                                     DM 00290
          RETURN;                                                            DM 00300
          END;                                                               DM 00310
        DIMLIST = DIMLIST || VARID || '|';                                   DM 00320
        RETURN;                                                              DM 00330
        END DIMENIT;                                                         DM 00340
```

```
/*  DO (DOSTMT)   20 MAR 70  THESIS VERSION  D. R. LITTLE  */                DO 00010
DOSTMT: PROC;                                                                DO 00020
        DCL LOOPNOS       CHAR(99)VAR    EXT,                              DO 00030
            NUMRLS        CHAR(10)       EXT,                              DO 00040
            STMT          CHAR(1320)VAR  EXT,                              DO 00050
            LOOPNO        CHAR(5)VAR,                                      DO 00060
            CONTNO        FIXED          EXT,                              DO 00070
            STRT          FIXED,                                           DO 00080
            SQZBLNK       ENTRY(FIXED,FIXED);                              DO 00090
        I = 0;                                                               DO 00100
D10:    IF INDEX(NUMRLS, SUBSTR(STMT, I+3, 1))=10 THEN DO;                  DO 00110
          I = I + 1;                                                         DO 00120
          GO TO D10;                                                         DO 00130
          END;                                                               DO 00140
        STRT = I + 3;                                                        DO 00150
        I = 1;                                                               DO 00160
D20:    IF INDEX(NUMRLS, SUBSTR(STMT, STRT+I, 1))>0 THEN DO;                 DO 00170
          I = I + 1;                                                         DO 00180
          IF I=6 THEN DO;                                                    DO 00190
            CALL ERROR ('DO-01: STMT NUMBER EXCEEDS 5 DIGITS.');             DO 00200
            RETURN;                                                          DO 00210
            END;                                                             DO 00220
          GO TO D20;                                                         DO 00230
          END;                                                               DO 00240
        LOOPNO = SUBSTR(STMT, STRT, I);                                      DO 00250
        IF LENGTH(LOOPNOS)=0 THEN GO TO D30;                                 DO 00260
        IF LOOPNO||'.'=SUBSTR(LOOPNOS, 1, I+1) THEN GO TO D40;               DO 00270
D30:    IF LENGTH(LOOPNOS)+I>93 THEN DO;                                     DO 00280
          CALL ERROR ('DO-02: DESIGN PARAMETER EXCEEDED: LOOPNOS.');         DO 00290
          RETURN;                                                            DO 00300
          END;                                                               DO 00310
        LOOPNOS = LOOPNO || '.' || SUBSTR(CONTNO, 4, 5) || LOOPNOS;          DO 00320
        CONTNO = CONTNO - 1;                                                 DO 00330
D40:    STMT = 'DO ' || SUBSTR(LOOPNOS, I+2, 5) || ' ' ||                    DO 00340
               SUBSTR(STMT, STRT+I);                                         DO 00350
        CALL SQZBLNK (10, 1321);                                             DO 00360
        CALL NUCARD;                                                         DO 00370
        RETURN;                                                              DO 00380
        END DOSTMT;                                                          DO 00390
```

```
/*  E  (ERROR)    17 MAR 70  THESIS VERSION  Do Ro LITTLE  */                    E  00010
ERROR:  PROC (MESSAGE);                                                          E  00020
        DCL MESSAGE       CHAR(47),                                              E  00030
            NSLIST        FILE PRINT      EXT,                                   E  00040
            IHESARC       ENTRY(FIXED(31,0) BIN);                                E  00050
        PUT FILE (NSLIST) SKIP (2) EDIT ('***** ERROR IN ABOVE ',                E  00060
            'STATEMENT:  ', MESSAGE) (X(10), A, A, A);                           E  00070
        PUT FILE (NSLIST) SKIP(3) EDIT (' ') (A);                                E  00080
        CALL IHESARC (8);                                                        E  00090
        RETURN;                                                                  E  00100
        END ERROR;                                                               E  00110
```

```
/*  EN (ENDSTMT)  17 MAR 70  THESIS VERSION  D. R. LITTLE  */                    EN 00010
ENDSTMT:PROC;                                                                    EN 00020
        DCL ACTFILE       CHAR(4)          EXT,                                EN 00030
            ALPHBET       CHAR(27)         EXT,                                EN 00040
            DIMSTMT       CHAR(1320)VAR    EXT,                                EN 00050
            EXPSTMT       CHAR(22)VAR      EXT,                                EN 00060
            NSDIM(6)      CHAR(3)VAR       EXT,                                EN 00070
            PLUG(8)       CHAR(11)VAR      EXT,                                EN 00080
            STMT          CHAR(1320)VAR    EXT,                                EN 00090
            STMTNO        CHAR(5)          EXT,                                EN 00100
            CARD          CHAR(80),                                            EN 00110
            NSTYP         CHAR(1),                                             EN 00120
            PROGEND       BIT(1)           EXT,                                EN 00130
            I             FIXED,                                               EN 00140
            NSTN(6,2)     FIXED            EXT,                                EN 00150
            TYPN(6,2)     FIXED            EXT,                                EN 00160
            NSOUT         FILE OUTPUT      EXT,                                EN 00170
            NSWORK        FILE             EXT;                                EN 00180
        CALL DUPEIT;                                                             EN 00190
        PROGEND = '1'B;                                                          EN 00200
        CLOSE FILE (NSWORK);                                                     EN 00210
        OPEN FILE (NSWORK) INPUT;                                                EN 00220
        ACTFILE = 'OUT ';                                                        EN 00230
        STMTNO = '     ';                                                        EN 00240
        DO I=1 TO 6;                                                             EN 00250
          IF TYPN(I, 2)>0                                                        EN 00260
            THEN DO;                                                             EN 00270
              STMT = PLUG(I) || 'T$' || SUBSTR(I, 8, 1) || '(';                  EN 00280
              IF TYPN(I, 2)<10                                                   EN 00290
                THEN STMT = STMT || SUBSTR(TYPN(I, 2), 8, 1) || ')';             EN 00300
                ELSE STMT = STMT || SUBSTR(TYPN(I, 2), 7, 2) || ')';             EN 00310
              IF NSTN(I, 2)>0 THEN DO;                                           EN 00320
                NSTYP = SUBSTR(ALPHBET, I, 1);                                   EN 00330
                STMT = STMT || ', T$' || NSTYP || '(' || NSDIM(I) ||             EN 00340
                       ',';                                                      EN 00350
                IF NSTN(I, 2)<10                                                 EN 00360
                  THEN STMT = STMT || SUBSTR(NSTN(I, 2), 8, 1) || ')';           EN 00370
                  ELSE STMT = STMT || SUBSTR(NSTN(I, 2), 7, 2) || ')';           EN 00380
                END;                                                             EN 00390
              END;                                                               EN 00400
            ELSE IF NSTN(I, 2)>0 THEN DO;                                        EN 00410
              NSTYP = SUBSTR(ALPHBET, I, 1);                                     EN 00420
              STMT = PLUG(I) || 'T$' || NSTYP || '(' || NSDIM(I) ||              EN 00430
                     ',';                                                        EN 00440
              IF NSTN(I, 2)<10                                                   EN 00450
                THEN STMT = STMT || SUBSTR(NSTN(I, 2), 8, 1) || ')';             EN 00460
                ELSE STMT = STMT || SUBSTR(NSTN(I, 2), 7, 2) || ')';             EN 00470
              END;                                                               EN 00480
```

```
          IF TYPN(I, 2)>0 | NSTN(I, 2)>0 THEN CALL NUCARD;                    EN 00490
          END;                                                                EN 00500
        DO I=7 TO 8;                                                          EN 00510
          IF TYPN(I,2)>0 THEN DO;                                             EN 00520
            STMT = PLUG(I) || 'T$' || SUBSTR(I, 8, 1) || '(';                 EN 00530
            IF TYPN(I,2)<10                                                   EN 00540
              THEN STMT = STMT || SUBSTR(TYPN(I,2), 8, 1) || ')';             EN 00550
              ELSE STMT = STMT || SUBSTR(TYPN(I,2), 7, 2) || ')';             EN 00560
            CALL NUCARD;                                                      EN 00570
            END;                                                              EN 00580
          END;                                                                EN 00590
        IF EXPSTMT¬=(0)' ' THEN DO;                                           EN 00600
          STMT = EXPSTMT;                                                     EN 00610
          CALL NUCARD;                                                        EN 00620
          END;                                                                EN 00630
        IF DIMSTMT¬=(0)' ' THEN DO;                                           EN 00640
          STMT = DIMSTMT;                                                     EN 00650
          CALL NUCARD;                                                        EN 00660
          END;                                                                EN 00670
        ON ENDFILE (NSWORK) BEGIN;                                            EN 00680
          CLOSE FILE (NSWORK);                                                EN 00690
          OPEN FILE (NSWORK) OUTPUT;                                          EN 00700
          GO TO E20;                                                          EN 00710
          END;                                                                EN 00720
E10:    GET FILE (NSWORK) EDIT (CARD) (A(80));                                EN 00730
        PUT FILE (NSOUT) EDIT (CARD) (A(80));                                 EN 00740
        GO TO E10;                                                            EN 00750
E20:    RETURN;                                                               EN 00760
        END ENDSTMT;                                                          EN 00770
```

```
/*  EQ (EQUIVST)  17 MAR 70  THESIS VERSION  D. R. LITTLE  */                EQ 00010
EQUIVST:PROC;                                                                EQ 00020
        DCL  ALPHBET      CHAR(27)          EXT,                           EQ 00030
             EXPLIST      CHAR(2000)VAR     EXT,                           EQ 00040
             IMPCODE      CHAR(27)VAR       EXT,                           EQ 00050
             IMPNSL       CHAR(27)VAR       EXT,                           EQ 00060
             STMT         CHAR(1320)VAR     EXT,                           EQ 00070
             VARID        CHAR(6)VAR        EXT,                           EQ 00080
             CHAR1        CHAR(1),                                         EQ 00090
             SYMB         CHAR(1),                                         EQ 00100
             VARTYP       CHAR(1),                                         EQ 00110
             ERRFLG       BIT(1)            EXT,                           EQ 00120
             NSSTAT       BIT(1)            EXT,                           EQ 00130
             EXPCHK       FIXED,                                           EQ 00140
             NSTNO        FIXED,                                           EQ 00150
             NXTDLM       FIXED,                                           EQ 00160
             STRT         FIXED,                                           EQ 00170
             SYMLOC       FIXED,                                           EQ 00180
             SEARCH       ENTRY(CHAR(1320)VAR),                            EQ 00190
             SQZBLNK      ENTRY(FIXED,FIXED);                              EQ 00200
        CALL SQZBLNK (1, 1321);                                           EQ 00210
        CALL SEARCH (SUBSTR(STMT, 12));                                   EQ 00220
        IF NSSTAT='0'B THEN DO;                                           EQ 00230
          CALL DUPEIT;                                                    EQ 00240
          RETURN;                                                         EQ 00250
          END;                                                            EQ 00260
        STMT = STMT || ';';                                               EQ 00270
        STRT = 13;                                                        EQ 00280
E10:    IF INDEX(ALPHBET, SUBSTR(STMT, STRT, 1))=0 THEN DO;               EQ 00290
          CALL ERROR ('EQ-01: EQUIVALENCE STMT SYNTAX ERROR.');           EQ 00300
          RETURN;                                                         EQ 00310
          END;                                                            EQ 00320
        NXTDLM= 1321;                                                     EQ 00330
        DO SYMB='(',',',')',';';                                          EQ 00340
          SYMLOC = INDEX(SUBSTR(STMT, STRT), SYMB);                       EQ 00350
          IF SYMLOC=0                                                     EQ 00360
            THEN SYMLOC = 1321;                                           EQ 00370
            ELSE SYMLOC = SYMLOC + STRT - 1;                              EQ 00380
          IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                          EQ 00390
          END;                                                            EQ 00400
        IF NXTDLM-STRT>6 THEN DO;                                         EQ 00410
          CALL ERROR ('EQ-02: VAR IDENTIFIER HAS > 6 CHAR.');             EQ 00420
          RETURN;                                                         EQ 00430
          END;                                                            EQ 00440
        SYMB = SUBSTR(STMT, NXTDLM, 1);                                   EQ 00450
        VARID = SUBSTR(STMT, STRT, NXTDLM-STRT);                          EQ 00460
        EXPCHK = INDEX(EXPLIST, ')'||VARID||'(');                         EQ 00470
        IF EXPCHK>0 THEN DO;                                              EQ 00480
```

```
            VARTYP = SUBSTR(EXPLIST, EXPCHK+NXTDLM-STRT+2, 1);                    EQ 00490
            NSTNO = INDEX(ALPHBET, VARTYP);                                       EQ 00500
            GO TO E20;                                                            EQ 00510
            END;                                                                  EQ 00520
          CHAR1 = SUBSTR(VARID, 1, 1);                                            EQ 00530
          IF INDEX(IMPNSL, CHAR1)>0                                               EQ 00540
            THEN DO;                                                              EQ 00550
              NSTNO = INDEX(ALPHBET, SUBSTR(IMPCODE, INDEX(ALPHBET,               EQ 00560
                      CHAR1), 1));                                                EQ 00570
              CALL DIMENIT (VARID, NSTNO);                                        EQ 00580
              IF ERRFLG='1'B THEN DO;                                             EQ 00590
                ERRFLG = '0'B;                                                    EQ 00600
                RETURN;                                                           EQ 00610
                END;                                                              EQ 00620
              END;                                                                EQ 00630
            ELSE NSTNO = 0;                                                       EQ 00640
E20:      IF SYMB='(' THEN DO;                                                    EQ 00650
            IF NSTNO>0 THEN STMT = SUBSTR(STMT, 1, NXTDLM) || '1,' ||             EQ 00660
                SUBSTR(STMT, NXTDLM+1);                                           EQ 00670
            STRT = INDEX(SUBSTR(STMT, NXTDLM), ')') + NXTDLM;                     EQ 00680
            IF STRT=NXTDLM THEN DO;                                               EQ 00690
              CALL ERROR ('EQ-03: NO CLOSE PAREN FOUND. ');                       EQ 00700
              RETURN;                                                             EQ 00710
              END;                                                                EQ 00720
            NXTDLM = STRT;                                                        EQ 00730
            SYMB = SUBSTR(STMT, NXTDLM, 1);                                       EQ 00740
            END;                                                                  EQ 00750
          IF SYMB=',' THEN DO;                                                    EQ 00760
            STRT = NXTDLM + 1;                                                    EQ 00770
            GO TO E10;                                                            EQ 00780
            END;                                                                  EQ 00790
          IF SYMB=')' THEN DO;                                                    EQ 00800
            IF SUBSTR(STMT, NXTDLM+1, 2)=',(' THEN DO;                            EQ 00810
              STRT = NXTDLM + 3;                                                  EQ 00820
              GO TO E10;                                                          EQ 00830
              END;                                                                EQ 00840
            IF SUBSTR(STMT, NXTDLM+1, 1)=';' THEN DO;                             EQ 00850
              STMT = SUBSTR(STMT, 1, NXTDLM);                                     EQ 00860
              CALL NUCARD;                                                        EQ 00870
              RETURN;                                                             EQ 00880
              END;                                                                EQ 00890
            END;                                                                  EQ 00900
          CALL ERROR ('EQ-04: ILLEGAL CHAR FOLLOWS '')'' OR AT END. ');           EQ 00910
          RETURN;                                                                 EQ 00920
          END EQUIVST;                                                            EQ 00930
```

```
/*   EX (EXPRSN)   01 AUG 70  THESIS VERSION  D. R. LITTLE  */                EX 00010
EXPRSN: PROC:                                                                 EX 00020
        DCL  ALPHBET       CHAR(27)          EXT,                           EX 00030
             EXPLIST       CHAR(2000)VAR     EXT,                           EX 00040
             EXPR          CHAR(1320)VAR     EXT,                           EX 00050
             IMPCODE       CHAR(27)          EXT,                           EX 00060
             NSDIM(6)      CHAR(3)VAR        EXT,                           EX 00070
             NSFNS         CHAR(141)VAR      EXT,                           EX 00080
             NUMRLS        CHAR(10)          EXT,                           EX 00090
             OPSET(9)      CHAR(5)VAR        EXT,                           EX 00100
             RSLT          CHAR(30)VAR       EXT,                           EX 00110
             RSLTYP        CHAR(1)           EXT,                           EX 00120
             STMT          CHAR(1320)VAR     EXT,                           EX 00130
             STMTNO        CHAR(5)           EXT,                           EX 00140
             ALPHNUM       CHAR(37),                                        EX 00150
             ARGS          CHAR(100)VAR,                                    EX 00160
             CHARSET       CHAR(39),                                        EX 00170
             CHAR1         CHAR(1),                                         EX 00180
             CURCHR        CHAR(1),                                         EX 00190
             DLMLIST       CHAR(20)            INIT('(),:789123456+-*/NE:'),EX 00200
             DLMPRI        CHAR(20)            INIT('01112345555556677789'),EX 00210
             FNSTAK        CHAR(71)VAR,                                     EX 00220
             NXTCHR        CHAR(1),                                         EX 00230
             OPERAND       CHAR(30)VAR,                                     EX 00240
             OPID          CHAR(6)VAR,                                      EX 00250
             OPRND(2)      CHAR(30)VAR,                                     EX 00260
             OPRNDTYP      CHAR(1),                                         EX 00270
             OPRTR         CHAR(5)VAR,                                      EX 00280
             OPTYP(2)      CHAR(1),                                         EX 00290
             RSLTBL        CHAR(144),                                       EX 00300
             RPSTR         CHAR(100)VAR,                                    EX 00310
             STCK          CHAR(30)VAR,                                     EX 00320
             STCKTOP       CHAR(1),                                         EX 00330
             SYMB          CHAR(2)VAR,                                      EX 00340
             CALSTMT       BIT(1)            EXT,                           EX 00350
             ERRFLG        BIT(1)            EXT,                           EX 00360
             NSSTAT        BIT(1)            EXT,                           EX 00370
             NSTN(6,2)     FIXED             EXT,                           EX 00380
             TYPN(8,2)     FIXED             EXT,                           EX 00390
             CURPRI        FIXED,                                           EX 00400
             DLMNO         FIXED,                                           EX 00410
             ENDPR         FIXED,                                           EX 00420
             EXPCHK        FIXED,                                           EX 00430
             I             FIXED,                                           EX 00440
             NCOMMA        FIXED,                                           EX 00450
             NSTNO         FIXED,                                           EX 00460
             NXTBAR        FIXED,                                           EX 00470
             NXTCOMA       FIXED,                                           EX 00480
```

```
            NXTCLPR       FIXED,                                              EX 00490
            NXTDLM        FIXED,                                              EX 00500
            NXTOPPR       FIXED,                                              EX 00510
            OPIDLN        FIXED,                                              EX 00520
            OPT(2)        FIXED,                                              EX 00530
            OPR           FIXED,                                              EX 00540
            STCKLN        FIXED,                                              EX 00550
            STCKTPPR      FIXED,                                              EX 00560
            STRT          FIXED,                                              EX 00570
            SUBSCR        FIXED,                                              EX 00580
            SYMLCC        FIXED;                                              EX 00590
          RPSTR = (0)' ';                                                     EX 00600
          STCK = (0)' ';                                                      EX 00610
          FNSTAK = (0)' ';                                                    EX 00620
          NCOMMA = 0;                                                         EX 00630
          EXPR = EXPR || '>;';                                                EX 00640
          DO I=1 TO 9;                                                        EX 00650
            STRT = 1;                                                         EX 00660
E10:        OPR = INDEX(SUBSTR(EXPR, STRT), OPSET(I)) + STRT - 1;             EX 00670
            IF OPR>STRT-1 THEN DO;                                            EX 00680
              IF I<8                                                          EX 00690
                THEN EXPR = SUBSTR(EXPR, 1, OPR-1) || '>' || SUBSTR(I, 8,EX 00700
                     1) || SUBSTR(EXPR, OPR+4);                               EX 00710
                ELSE EXPR = SUBSTR(EXPR, 1, OPR-1) || '>' || SUBSTR(I, 8,EX 00720
                     1) || SUBSTR(EXPR, OPR+5);                               EX 00730
              IF I=9 THEN DO;                                                 EX 00740
                IF OPR=1 THEN GO TO E20;                                      EX 00750
                IF INDEX('(,', SUBSTR(EXPR, OPR-1, 1))>0 | INDEX('>7>8',      EX 00760
                     SUBSTR(EXPR, OPR-2, 2))>0 THEN GO TO E20;                EX 00770
                CALL ERROR ('EX-01: ''¬NOT¬'' IN ILLEGAL POSITION¬');         EX 00780
                GO TO E190;                                                   EX 00790
                END;                                                          EX 00800
E20:          STRT = OPR + 2;                                                 EX 00810
              GO TO E10;                                                      EX 00820
              END;                                                            EX 00830
            END;                                                              EX 00840
          STRT = 1;                                                           EX 00850
E30:      OPR = INDEX(SUBSTR(EXPR, STRT), '+') + STRT - 1;                    EX 00860
          IF OPR>STRT-1 THEN DO;                                              EX 00870
            IF OPR=1 THEN DO;                                                 EX 00880
              EXPR = SUBSTR(EXPR, 2);                                         EX 00890
              GO TO E30;                                                      EX 00900
              END;                                                            EX 00910
            IF INDEX('(,', SUBSTR(EXPR, OPR-1, 1))>0 | SUBSTR(EXPR,OPR-2,EX 00920
                1)='>' THEN DO;                                               EX 00930
              EXPR = SUBSTR(EXPR, 1, OPR-1) || SUBSTR(EXPR, OPR+1);           EX 00940
              STRT = OPR;                                                     EX 00950
              GO TO E30;                                                      EX 00960
```

```
            END;                                                                EX 00970
          STRT = OPR + 1;                                                       EX 00980
          GO TO E30;                                                            EX 00990
          END;                                                                  EX 01000
        STRT = 1;                                                               EX 01010
E40:    OPR = INDEX(SUBSTR(EXPR, STRT), '-') + STRT - 1;                        EX 01020
        IF OPR>STRT-1 THEN DO;                                                  EX 01030
          IF OPR=1 THEN DO;                                                     EX 01040
            EXPR = '>N'|| SUBSTR(EXPR, 2);                                      EX 01050
            GO TO E40;                                                          EX 01060
            END;                                                                EX 01070
          IF INDEX('(,', SUBSTR(EXPR, OPR-1, 1))>0 | SUBSTR(EXPR,OPR-2,EX 01080
              1)='>' THEN DO;                                                   EX 01090
            EXPR = SUBSTR(EXPR, 1,OPR-1) || '>N' || SUBSTR(EXPR,OPR+1);EX 01100
            STRT = OPR;                                                         EX 01110
            GO TO E40;                                                          EX 01120
            END;                                                                EX 01130
          STRT = OPR + 1;                                                       EX 01140
          GO TO E40;                                                            EX 01150
          END;                                                                  EX 01160
        STRT = 1;                                                               EX 01170
E50:    OPR = INDEX(SUBSTR(EXPR, STRT), '**') + STRT - 1;                       EX 01180
        IF OPR>STRT-1 THEN DO;                                                  EX 01190
          EXPR = SUBSTR(EXPR, 1, OPR-1) || '>E' || SUBSTR(EXPR, OPR+2);EX 01200
          STRT = OPR + 2;                                                       EX 01210
          GO TO E50;                                                            EX 01220
          END;                                                                  EX 01230
        STRT = 1;                                                               EX 01240
        NXTCHR = SUBSTR(EXPR, 1, 1);                                            EX 01250
        RSLTBL = '121156ABCDEF222266ABCDEF123456ABCDEF124456ABCDEF' ||          EX 01260
                 '565556ABCDEF666666ABCDEFAAAAAAABAAEFBBBBBBBBBBEF' ||          EX 01270
                 'CCCCCCABCDEFDDDDDDABDDEF' || (11)'E' || (13)'F';              EX 01280
        ALPHNUM = ALPHBET || NUMRLS;                                            EX 01290
        CHARSET = ALPHNUM || '.(';                                              EX 01300
        NXTDLM = 1;                                                             EX 01310
        IF NXTCHR = '>' THEN DO;                                                EX 01320
          IF SUBSTR(EXPR, 2, 1)='N' | SUBSTR(EXPR, 2, 1)='9' THEN GO TOEX 01330
            E110;                                                               EX 01340
          END;                                                                  EX 01350
        IF INDEX(ALPHBET, NXTCHR)>0 THEN GO TO E60;                             EX 01360
        IF INDEX(NUMRLS, NXTCHR)>0 THEN GO TO E80;                              EX 01370
        IF NXTCHR='(' THEN GO TO E100;                                          EX 01380
        IF NXTCHR='.' THEN GO TO E70;                                           EX 01390
        CALL ERROR ('EX-02: 1ST CHAR OF EXPRESSION IS ILLEGAL.');               EX 01400
        GO TO E190;                                                             EX 01410
E60:    NXTDLM = 1321;                                                          EX 01420
        DO SYMB='+','-','*','/','(',')',',','>';                                EX 01430
          SYMLOC = INDEX(SUBSTR(EXPR, STRT), SYMB) + STRT - 1;                  EX 01440
```

```
      IF SYMLOC=STRT-1 THEN SYMLOC = 1321;                                  EX 01450
      IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                                EX 01460
      END;                                                                  EX 01470
    IF NXTDLM-STRT>6 THEN DO;                                               EX 01480
      CALL ERROR ('EX-03: VAR OR FUNC IDENTIFIER HAS > 6 CHAR.');           EX 01490
      GO TO E190;                                                           EX 01500
      END;                                                                  EX 01510
    IF SUBSTR(EXPR, NXTDLM, 1)='(' THEN DO;                                 EX 01520
      IF INDEX(NSFNS, '|'||SUBSTR(EXPR, STRT, NXTDLM-STRT)||'|')>0          EX 01530
          THEN DO;                                                          EX 01540
        IF LENGTH(FNSTAK)>65 THEN DO;                                       EX 01550
          CALL ERROR ('EX-04: DESIGN PARAMETER EXCEEDED: FNSTAK.');         EX 01560
          GO TO E190;                                                       EX 01570
          END;                                                              EX 01580
        FNSTAK = SUBSTR(EXPR, STRT, NXTDLM-STRT) || '|' || FNSTAK;          EX 01590
        EXPR = SUBSTR(EXPR, 1, STRT-1) || ':' || SUBSTR(EXPR,               EX 01600
            NXTDLM);                                                        EX 01610
        NXTDLM = STRT;                                                      EX 01620
        CURCHR = ':';                                                       EX 01630
        GO TO E120;                                                         EX 01640
        END;                                                                EX 01650
      CALL PARLOC (EXPR, NXTDLM, ENDPR);                                    EX 01660
      IF ERRFLG='1'B THEN RETURN;                                           EX 01670
      CALL SEARCH (SUBSTR(EXPR, NXTDLM, ENDPR-NXTDLM+1));                   EX 01680
      IF NSSTAT='1'B THEN DO;                                               EX 01690
        CALL ERROR ('EX-05: NONSTD IN SUBSCRPTS OR FUNC ARG.');             EX 01700
        GO TO E190;                                                         EX 01710
        END;                                                                EX 01720
      NXTDLM = ENDPR + 1;                                                   EX 01730
      END;                                                                  EX 01740
    IF LENGTH(RPSTR)+NXTDLM-STRT+1>100 THEN DO;                             EX 01750
      CALL ERROR ('EX-06: DESIGN PARAMETER EXCEEDED: RPSTR.');              EX 01760
      GO TO E190;                                                           EX 01770
      END;                                                                  EX 01780
    RPSTR = SUBSTR(EXPR, STRT, NXTDLM-STRT) || '|' || RPSTR;                EX 01790
    GO TO E110;                                                             EX 01800
E70:IF INDEX(NUMRLS, SUBSTR(EXPR, STRT+1, 1))>0 THEN GO TO E80;             EX 01810
    NXTDLM = INDEX(SUBSTR(EXPR, STRT+1), '.');                              EX 01820
    IF NXTDLM¬=5 & NXTDLM¬=6 THEN DO;                                       EX 01830
      CALL ERROR ('EX-07: ILLEGAL CHARS FOLLOWING ''.''.');                 EX 01840
      GO TO E190;                                                           EX 01850
      END;                                                                  EX 01860
    NXTDLM = NXTDLM + STRT + 1;                                             EX 01870
    IF INDEX(')',', SUBSTR(EXPR, NXTDLM, 1))=0 & INDEX('>;>7>8',           EX 01880
        SUBSTR(EXPR, NXTDLM, 2))=0 THEN DO;                                 EX 01890
      CALL ERROR ('EX-08: ILLEGAL CHAR FOLLOWS LOG CONST.');                EX 01900
      GO TO E190;                                                           EX 01910
      END;                                                                  EX 01920
```

```
          IF LENGTH(RPSTR)+NXTDLM-STRT+1>100 THEN DO;                          EX 01930
            CALL ERROR ('EX-09: DESIGN PARAMETER EXCEEDED: RPSTR.');           EX 01940
            GO TO E190;                                                        EX 01950
            END;                                                               EX 01960
          RPSTR = SUBSTR(EXPR, STRT, NXTDLM-STRT) || '|' || RPSTR;            EX 01970
          GO TO E110;                                                          EX 01980
E80:      STRT1 = STRT;                                                        EX 01990
E90:      NXTDLM = 1321;                                                       EX 02000
          DO SYMB='+','-','*','/',')',',','>';                                 EX 02010
            SYMLOC = INDEX(SUBSTR(EXPR, STRT1), SYMB) + STRT1 - 1;             EX 02020
            IF SYMLOC=STRT1-1 THEN SYMLOC = 1321;                              EX 02030
            IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                             EX 02040
            END;                                                               EX 02050
          IF (SUBSTR(EXPR, NXTDLM, 1)='+' | SUBSTR(EXPR, NXTDLM, 1)='-')&EX 02060
              (SUBSTR(EXPR, NXTDLM-1, 1)='E' | SUBSTR(EXPR, NXTDLM-1, 1)EX 02070
              ='D') THEN DO;                                                   EX 02080
            STRT1 = NXTDLM + 1;                                                EX 02090
            GO TO E90;                                                         EX 02100
            END;                                                               EX 02110
          IF LENGTH(RPSTR)+NXTDLM-STRT+1>100 THEN DO;                          EX 02120
            CALL ERROR ('EX-10: DESIGN PARAMETER EXCEEDED: RPSTR.');           EX 02130
            GO TO E190;                                                        EX 02140
            END;                                                               EX 02150
          RPSTR = SUBSTR(EXPR, STRT, NXTDLM-STRT) || '|' || RPSTR;            EX 02160
          GO TO E110;                                                          EX 02170
E100:     IF INDEX(NUMRLS||'.', SUBSTR(EXPR, STRT+1, 1))=0 THEN GO TO          EX 02180
              E110;                                                            EX 02190
          NXTCOMA = INDEX(SUBSTR(EXPR, STRT+1), ',') + STRT;                   EX 02200
          IF NXTCOMA=STRT THEN GO TO E110;                                     EX 02210
          NXTCLPR = INDEX(SUBSTR(EXPR, STRT+1), ')') + STRT;                   EX 02220
          IF NXTCLPR=STRT THEN DO;                                             EX 02230
            CALL ERROR ('EX-11: NO CLOSE PAREN FOUND.');                       EX 02240
            GO TO E190;                                                        EX 02250
            END;                                                               EX 02260
          IF NXTCLPR<NXTCOMA THEN GO TO E110;                                  EX 02270
          NXTOPPR = INDEX(SUBSTR(EXPR, STRT+1), '(') + STRT;                   EX 02280
          IF NXTOPPR>STRT & NXTOPPR<NXTCLPR THEN GO TO E110;                   EX 02290
          IF LENGTH(RPSTR)+NXTCLPR-STRT+2>100 THEN DO;                         EX 02300
            CALL ERROR ('EX-12: DESIGN PARAMETER EXCEEDED: RPSTR.');           EX 02310
            GO TO E190;                                                        EX 02320
            END;                                                               EX 02330
          RPSTR = SUBSTR(EXPR, STRT, NXTCLPR-STRT+1) || '|' || RPSTR;         EX 02340
          NXTDLM = NXTCLPR + 1;                                                EX 02350
          IF INDEX('+-*/),', SUBSTR(EXPR, NXTDLM, 1))=0 & INDEX(               EX 02360
              '>E>;', SUBSTR(EXPR, NXTDLM, 2))=0 THEN DO;                      EX 02370
            CALL ERROR ('EX-13: ILLEGAL CHAR FOLLOWS CMPLX CONSTANT.');        EX 02380
            GO TO E190;                                                        EX 02390
            END;                                                               EX 02400
```

```
E110:   CURCHR = SUBSTR(EXPR, NXTDLM, 1);                                          EX 02410
        IF CURCHR='>' THEN DO;                                                     EX 02420
          EXPR = SUBSTR(EXPR, 1, NXTDLM-1) || SUBSTR(EXPR, NXTDLM+1);              EX 02430
          CURCHR = SUBSTR(EXPR, NXTDLM, 1);                                        EX 02440
          IF CURCHR=';' THEN GO TO E120;                                           EX 02450
          END;                                                                     EX 02460
        DLMNO = INDEX(DLMLIST, CURCHR);                                            EX 02470
        NXTCHR = SUBSTR(EXPR, NXTDLM+1, 1);                                        EX 02480
        IF DLMNO¬=2 & INDEX(CHARSET, NXTCHR)>0 THEN GO TO E120;                    EX 02490
        IF DLMNO=2 & INDEX('+-*/),>', NXTCHR)>0 THEN GO TO E120;                   EX 02500
        IF (DLMNO=1 | DLMNO=3 | DLMNO=5 | DLMNO=6) & INDEX('>N>9',                 EX 02510
            SUBSTR(EXPR, NXTDLM+1, 2))>0 THEN GO TO E120;                          EX 02520
        IF DLMNO>4 & DLMNO<14 & SUBSTR(EXPR, NXTDLM+1, 2)='>N' THEN GO             EX 02530
            TO E120;                                                               EX 02540
        CALL ERROR ('EX-14: ILLEGAL CHAR FOLLOWS DELIMITER.');                     EX 02550
        GO TO E190;                                                                EX 02560
E120:   CURPRI = SUBSTR(DLMPRI, INDEX(DLMLIST, CURCHR), 1);                        EX 02570
E130:   STCKLN = LENGTH(STCK);                                                     EX 02580
        IF STCKLN=0 THEN DO;                                                       EX 02590
          IF CURCHR=')' THEN DO;                                                   EX 02600
            CALL ERROR ('EX-15: UNMATCHED CLOSE PAREN FOUND.');                    EX 02610
            GO TO E190;                                                            EX 02620
            END;                                                                   EX 02630
          IF CURCHR=';' THEN GO TO E170;                                           EX 02640
          STCK = CURCHR;                                                           EX 02650
          GO TO E140;                                                              EX 02660
          END;                                                                     EX 02670
        STCKTOP = SUBSTR(STCK, 1, 1);                                              EX 02680
        IF CURCHR='(' | (CURCHR=',' & STCKTOP=',') THEN DO;                        EX 02690
          IF STCKLN=30 THEN DO;                                                    EX 02700
            CALL ERROR ('EX-16: DESIGN PARAMETER EXCEEDED: STCK.');                EX 02710
            GO TO E190;                                                            EX 02720
            END;                                                                   EX 02730
          STCK = CURCHR || STCK;                                                   EX 02740
          GO TO E140;                                                              EX 02750
          END;                                                                     EX 02760
        STCKTPPR = SUBSTR(DLMPRI, INDEX(DLMLIST, STCKTOP), 1);                     EX 02770
        IF CURPRI>STCKTPPR                                                         EX 02780
          THEN DO;                                                                 EX 02790
            IF CURCHR=')' & STCKTOP='(' THEN DO;                                   EX 02800
              STCK = SUBSTR(STCK, 2);                                              EX 02810
              GO TO E140;                                                          EX 02820
              END;                                                                 EX 02830
            IF CURCHR=';' & STCKTOP='(' THEN DO;                                   EX 02840
              CALL ERROR ('EX-17: UNMATCHED OPEN PAREN FOUND.');                   EX 02850
              GO TO E190;                                                          EX 02860
              END;                                                                 EX 02870
            IF STCKLN=30 THEN DO;                                                  EX 02880
```

```
            CALL ERROR ('EX-18: DESIGN PARAMETER EXCEEDED: STCK.');   EX 02890
            GO TO E190;                                               EX 02900
            END;                                                      EX 02910
          STCK = CURCHR || STCK;                                      EX 02920
          END;                                                        EX 02930
        ELSE DO;                                                      EX 02940
          STCK = SUBSTR(STCK, 2);                                     EX 02950
          GO TO E150;                                                 EX 02960
          END;                                                        EX 02970
E140:   NXTDLM = NXTDLM + 1;                                          EX 02980
        NXTCHR = SUBSTR(EXPR, NXTDLM, 1);                             EX 02990
        STRT = NXTDLM;                                                EX 03000
        IF INDEX(ALPHBET, NXTCHR)>0 THEN GO TO E60;                   EX 03010
        IF INDEX(NUMRLS, NXTCHR)>0 THEN GO TO E80;                    EX 03020
        IF NXTCHR='(' THEN GO TO E100;                                EX 03030
        IF NXTCHR='.' THEN GO TO E70;                                 EX 03040
        GO TO E110;                                                   EX 03050
E150:   IF STCKTCP=':' THEN DO;                                       EX 03060
          ARGS = (0)' ';                                              EX 03070
          DO I=1 TO NCOMMA+1;                                         EX 03080
            NXTBAR = INDEX(RPSTR, '|');                               EX 03090
            IF NXTBAR>31 THEN DO;                                     EX 03100
              CALL ERROR('EX-19: DESIGN PARAMETER EXCEEDED: OPERAND.');EX 03110
              GO TO E190;                                             EX 03120
              END;                                                    EX 03130
            OPERAND = SUBSTR(RPSTR, 1, NXTBAR-1);                     EX 03140
            CALL OPTYPE;                                              EX 03150
            IF LENGTH(OPERAND)+NXTBAR>100 THEN DO;                    EX 03160
              CALL ERROR ('EX-20: DESIGN PARAMETER EXCEEDED: ARGS.');   EX 03170
              GO TO E190;                                             EX 03180
              END;                                                    EX 03190
            ARGS = OPERAND || ', ' || ARGS;                           EX 03200
            RPSTR = SUBSTR(RPSTR, NXTBAR+1);                          EX 03210
            END;                                                      EX 03220
          NXTBAR = INDEX(FNSTAK, '|');                                EX 03230
          OPERAND = SUBSTR(FNSTAK, 1, NXTBAR-1);                      EX 03240
          CALL OPTYPE;                                                EX 03250
          IF NXTBAR=7 THEN OPERAND = SUBSTR(OPERAND, 1, 5);           EX 03260
          NXTOPPR = INDEX(OPERAND, '(');                              EX 03270
          IF NXTOPPR>0 THEN OPERAND = SUBSTR(OPERAND, 1, NXTOPPR-1);  EX 03280
          FNSTAK = SUBSTR(FNSTAK, NXTBAR+1);                          EX 03290
          NSTNO = INDEX(ALPHBET, OPRNDTYP);                           EX 03300
          NSTN(NSTNO, 1) = NSTN(NSTNO, 1) + 1;                        EX 03310
          RSLT = 'T$' || OPRNDTYP || '(1,' || SUBSTR(NSTN(NSTNO,1),   EX 03320
                 7, 2) || ')';                                        EX 03330
          IF NSTN(NSTNO, 1)<10 THEN RSLT = SUBSTR(RSLT, 1, 6) ||      EX 03340
              SUBSTR(RSLT, 8);                                        EX 03350
          IF LENGTH(RPSTR)+LENGTH(RSLT)+1>100 THEN DO;                EX 03360
```

```
      CALL ERROR ('EX-21: DESIGN PARAMETER EXCEEDED: RPSTR.');    EX 03370
      GO TO E190;                                                 EX 03380
      END;                                                        EX 03390
    RPSTR = RSLT || '|' || RPSTR;                                 EX 03400
    STMT = 'CALL $' || OPERAND || ' (' || RSLT || ', ' ||         EX 03410
           SUBSTR(ARGS, 1, LENGTH(ARGS)-2) || ')';                EX 03420
    CALL NUCARD;                                                  EX 03430
    STMTNO = '     ';                                             EX 03440
    NCOMMA = 0;                                                   EX 03450
    GO TO E130;                                                   EX 03460
    END;                                                          EX 03470
  IF STCKTCP=',' THEN DO;                                         EX 03480
    NCOMMA = NCOMMA + 1;                                          EX 03490
    GO TO E130;                                                   EX 03500
    END;                                                          EX 03510
  IF STCKTCP='9' | STCKTOP='N'                                    EX 03520
    THEN NOPS = 1;                                                EX 03530
    ELSE NOPS = 2;                                                EX 03540
  OPRTR = STCKTOP;                                                EX 03550
  DO I=NOPS TO 1 BY -1;                                           EX 03560
    NXTBAR = INDEX(RPSTR, '|');                                   EX 03570
    IF NXTBAR>31 THEN DO;                                         EX 03580
      CALL ERROR ('EX-22: DESIGN PARAMETER EXCEEDED: OPERAND.');  EX 03590
      GO TO E190;                                                 EX 03600
      END;                                                        EX 03610
    OPRND(I) = SUBSTR(RPSTR, 1, NXTBAR-1);                        EX 03620
    OPERAND = OPRND(I);                                           EX 03630
    CALL OPTYPE;                                                  EX 03640
    OPRND(I) = OPERAND;                                           EX 03650
    IF OPRNDTYP='7' | OPRNDTYP='8' THEN IF INDEX('789', OPRTR)=0  EX 03660
        THEN DO;                                                  EX 03670
      CALL ERROR ('EX-23: LOG VAR FOUND IN ARITH EXPR.');         EX 03680
      GO TO E190;                                                 EX 03690
      END;                                                        EX 03700
    IF INDEX('789',OPRTR)>0 THEN IF OPRNDTYP¬='7' & OPRNDTYP¬='8' EX 03710
        THEN DO;                                                  EX 03720
      CALL ERROR ('EX-24: LOG EXPR CONTAINS NONLOG VAR.');        EX 03730
      GO TO E190;                                                 EX 03740
      END;                                                        EX 03750
    OPTYP(I) = OPRNDTYP;                                          EX 03760
    RPSTR = SUBSTR(RPSTR, NXTBAR+1);                              EX 03770
    END;                                                          EX 03780
  IF OPRTR='N' THEN DO;                                           EX 03790
    RSLTYP = OPTYP(1);                                            EX 03800
    GO TO E160;                                                   EX 03810
    END;                                                          EX 03820
  IF INDEX('123456789', OPRTR)>0 THEN DO;                         EX 03830
    RSLTYP = '7';                                                 EX 03840
```

```
            GO TO E160;                                                          EX 03850
            END;                                                                 EX 03860
          DO I=1 TO 2;                                                           EX 03870
            IF INDEX(NUMRLS, OPTYP(I))>0                                         EX 03880
              THEN OPT(I) = OPTYP(I);                                            EX 03890
              ELSE OPT(I) = INDEX(ALPHBET, OPTYP(I)) + 6;                        EX 03900
            END;                                                                 EX 03910
          RSLTYP = SUBSTR(RSLTBL, 12*(OPT(1)-1)+OPT(2), 1);                      EX 03920
E160:     NSTNO = INDEX(ALPHBET, RSLTYP);                                        EX 03930
          IF NSTNO>0                                                             EX 03940
            THEN DO;                                                             EX 03950
              NSTN(NSTNO, 1) = NSTN(NSTNO, 1) + 1;                               EX 03960
              RSLT = 'T$' || RSLTYP || '(1,' || SUBSTR(NSTN(NSTNO,1),7,2)EX 03970
                     || ')';                                                     EX 03980
              IF NSTN(NSTNO, 1)<10 THEN RSLT = SUBSTR(RSLT, 1, 6) ||         EX 03990
                 SUBSTR(RSLT, 8);                                              EX 04000
              END;                                                               EX 04010
            ELSE DO;                                                             EX 04020
              TYPNO = RSLTYP;                                                    EX 04030
              TYPN(TYPNO, 1) = TYPN(TYPNO, 1) + 1;                               EX 04040
              RSLT = 'T$' || RSLTYP || '(' || SUBSTR(TYPN(TYPNO,1),7, 2)  EX 04050
                     || ')';                                                     EX 04060
              IF TYPN(TYPNO, 1)<10 THEN RSLT = SUBSTR(RSLT, 1, 4) ||         EX 04070
                 SUBSTR(RSLT, 6);                                              EX 04080
              END;                                                               EX 04090
          IF LENGTH(RPSTR)+LENGTH(RSLT)+1>100 THEN DO;                           EX 04100
            CALL ERROR ('EX-25: DESIGN PARAMETER EXCEEDED: RPSTR.');             EX 04110
            GO TO E190;                                                          EX 04120
            END;                                                                 EX 04130
          RPSTR = RSLT || '|' || RPSTR;                                          EX 04140
          IF INDEX(ALPHBET, OPTYP(1))>0 | INDEX(ALPHBET, OPTYP(2))>0             EX 04150
            THEN DO;                                                             EX 04160
              IF OPRTR='+' THEN OPRTR = 'A';                                     EX 04170
              IF OPRTR='-' THEN OPRTR = 'S';                                     EX 04180
              IF OPRTR='*' THEN OPRTR = 'M';                                     EX 04190
              IF OPRTR='/' THEN OPRTR = 'D';                                     EX 04200
              IF OPRTR='N'                                                       EX 04210
                THEN STMT = 'CALL N$' || RSLTYP || OPTYP(1) || 'N  (' ||  EX 04220
                            RSLT || ', ' || OPRND(1) || ')';                     EX 04230
                ELSE STMT = 'CALL N$' || RSLTYP || OPTYP(1) || OPRTR ||  EX 04240
                            OPTYP(2) || ', (' || RSLT || ', ' || OPRND(1)  EX 04250
                            || ', ' || OPRND(2) || ')';                          EX 04260
              END;                                                               EX 04270
            ELSE DO;                                                             EX 04280
              IF INDEX(NUMRLS, OPRTR)>0 THEN OPRTR = OPSET(FIXED(OPRTR));EX 04290
              IF OPRTR='E' THEN OPRTR='**';                                      EX 04300
              IF OPRTR='N' | OPRTR='.NOT.'                                       EX 04310
                THEN IF OPRTR='N'                                                EX 04320
```

```
                  THEN STMT = RSLT || ' = - ' || OPRND(1);                    EX 04330
                  ELSE STMT = RSLT || ' = °NOT° ' || OPRND(1);                EX 04340
               ELSE STMT = RSLT || ' = ' || OPRND(1) || ' ' ||                EX 04350
                           OPRTR || ' ' || OPRND(2);                          EX 04360
            END;                                                              EX 04370
         CALL NUCARD;                                                         EX 04380
         STMTNO = '     ';                                                    EX 04390
         GO TO E130;                                                          EX 04400
E170:    NXTBAR = INDEX(RPSTR, '|');                                          EX 04410
         IF INDEX(SUBSTR(RPSTR, NXTBAR+1), '|')>0 THEN DO;                    EX 04420
           CALL ERROR ('EX-26: EXPR SYNTAX IN ERROR°');                       EX 04430
           GO TO E190;                                                        EX 04440
           END;                                                               EX 04450
         IF NXTBAR>31 THEN DO;                                                EX 04460
           CALL ERROR ('EX-27: DESIGN PARAMETER EXCEEDED: OPERAND°');         EX 04470
           GO TO E190;                                                        EX 04480
           END;                                                               EX 04490
         OPERAND = SUBSTR(RPSTR, 1, NXTBAR-1);                                EX 04500
         CALL CPTYPE;                                                         EX 04510
         RSLT = OPERAND;                                                      EX 04520
         RSLTYP = OPRNDTYP;                                                   EX 04530
         IF CALSTMT='1'B THEN RETURN;                                         EX 04540
E180:    DO I=1 TO 8;                                                         EX 04550
           IF I<7 THEN DO;                                                    EX 04560
             IF NSTN(I,1)>NSTN(I,2) THEN NSTN(I,2) = NSTN(I,1);               EX 04570
             NSTN(I,1) = 0;                                                   EX 04580
             END;                                                             EX 04590
           IF TYPN(I,1)>TYPN(I,2) THEN TYPN(I,2) = TYPN(I,1);                 EX 04600
           TYPN(I,1) = 0;                                                     EX 04610
           END;                                                               EX 04620
         RETURN;                                                              EX 04630
E190;    ERRFLG = '1'B;                                                       EX 04640
         GO TO E180;                                                          EX 04650
OPTYPE:  PROC;                                                                EX 04660
         IF LENGTH(OPERAND)>1 THEN IF SUBSTR(OPERAND, 1,2)='T$' THEN DO;      EX 04670
           OPRNDTYP = SUBSTR(OPERAND, 3, 1);                                  EX 04680
           RETURN;                                                            EX 04690
           END;                                                               EX 04700
         CHAR1 = SUBSTR(OPERAND, 1, 1);                                       EX 04710
         IF INDEX(ALPHBET, CHAR1)=0 THEN GO TO E200;                          EX 04720
         SUBSCR = INDEX(OPERAND, '(');                                        EX 04730
         IF SUBSCR=0                                                          EX 04740
           THEN OPID = OPERAND;                                               EX 04750
           ELSE OPID = SUBSTR(OPERAND, 1, SUBSCR-1);                          EX 04760
         OPIDLN = LENGTH(OPID);                                               EX 04770
         EXPCHK = INDEX(EXPLIST, ')'||OPID||'(');                             EX 04780
         IF EXPCHK>0                                                          EX 04790
           THEN OPRNDTYP = SUBSTR(EXPLIST, EXPCHK+OPIDLN+2, 1);               EX 04800
```

```
          ELSE DO;                                                          EX 04810
            OPRNDTYP = SUBSTR(IMPCODE, INDEX(ALPHBET,                       EX 04820
                       SUBSTR(OPID, 1, 1)), 1);                             EX 04830
            NSTNO = INDEX(ALPHBET, OPRNDTYP);                               EX 04840
            IF NSTNO>0 THEN CALL DIMENIT (OPID, NSTNO);                     EX 04850
            IF ERRFLG='1'B THEN RETURN;                                     EX 04860
            END;                                                            EX 04870
          IF INDEX(ALPHBET, OPRNDTYP)>0 THEN IF SUBSCR=0                    EX 04880
            THEN OPERAND = OPERAND || '(1)';                                EX 04890
            ELSE DO;                                                        EX 04900
              IF LENGTH(OPERAND)>28 THEN DO;                                EX 04910
                CALL ERROR('EX-28: DESIGN PARAMETER EXCEEDED: OPERAND.');   EX 04920
                GO TO E190;                                                 EX 04930
                END;                                                        EX 04940
              OPERAND = SUBSTR(OPERAND, 1, SUBSCR) || '1,' ||               EX 04950
                        SUBSTR(OPERAND, SUBSCR+1);                          EX 04960
              END;                                                          EX 04970
          RETURN;                                                           EX 04980
E200:     IF CHAR1='.' & INDEX(NUMRLS, SUBSTR(OPERAND, 2, 1))=0 THEN DO;    EX 04990
            OPRNDTYP = '8';                                                 EX 05000
            RETURN;                                                         EX 05010
            END;                                                            EX 05020
          IF INDEX(NUMRLS, CHAR1)>0 | CHAR1='.' THEN DO;                    EX 05030
            IF INDEX(OPERAND, 'E')>0 THEN DO;                               EX 05040
              OPRNDTYP = '1';                                               EX 05050
              RETURN;                                                       EX 05060
              END;                                                          EX 05070
            IF INDEX(OPERAND, 'D')>0 THEN DO;                               EX 05080
              OPRNDTYP = '2';                                               EX 05090
              RETURN;                                                       EX 05100
              END;                                                          EX 05110
            IF INDEX(OPERAND, '.')=0 THEN DO;                               EX 05120
              OPRNDTYP = '4';                                               EX 05130
              RETURN;                                                       EX 05140
              END;                                                          EX 05150
            OPIDLN = LENGTH(OPERAND);                                       EX 05160
            IF OPIDLN>8                                                     EX 05170
              THEN OPRNDTYP = '2';                                          EX 05180
              ELSE OPRNDTYP = '1';                                          EX 05190
            RETURN;                                                         EX 05200
            END;                                                            EX 05210
          IF CHAR1='(' THEN DO;                                             EX 05220
            IF INDEX(OPERAND, 'D')>0 THEN DO;                               EX 05230
              OPRNDTYP = '6';                                               EX 05240
              RETURN;                                                       EX 05250
              END;                                                          EX 05260
            OPRNDTYP = '5';                                                 EX 05270
            RETURN;                                                         EX 05280
```

```
EX 05290
EX 05300
EX 05310
```

```
END;
END OPTYPE;
END EXPRSN;
```

```
/*  F  (FUNCST)   28 JAN 70  THESIS VERSION  D. R. LITTLE  */                F  00010
FUNCST: PROC;                                                                F  00020
        DCL  ALPHBET        CHAR(27)          EXT,                           F  00030
             DIMLIST        CHAR(1000)VAR     EXT,                           F  00040
             EXPLIST        CHAR(2000)VAR     EXT,                           F  00050
             EXPNSL         CHAR(141)VAR      EXT,                           F  00080
             EXPSTMT        CHAR(22)VAR       EXT,                           F  00060
             NSDIM(6)       CHAR(3)VAR        EXT,                           F  00070
             NUMRLS         CHAR(10)          EXT,                           F  00090
             PLUG(8)        CHAR(11)VAR       EXT,                           F  00100
             STMT           CHAR(1320)VAR     EXT,                           F  00110
             CHAR1          CHAR(1),                                         F  00120
             CODE           CHAR(1),                                         F  00130
             FIRSTCHR       CHAR(6)                 INIT('CDILNR'),          F  00140
             NSTYPE         CHAR(1),                                         F  00150
             VARIC          CHAR(6)VAR,                                      F  00160
             NSTNO          FIXED,                                           F  00170
             NXTDLM         FIXED,                                           F  00180
             PAREN          FIXED,                                           F  00190
             STAR           FIXED,                                           F  00200
             STRT           FIXED,                                           F  00210
             NSTDST         ENTRY(FIXED,,,),                                 F  00220
             SQZBLNK        ENTRY(FIXED,FIXED),                              F  00230
             LBLTBL(0:6)    LABEL                   INIT(F80,F10,F20,F30,F40,F  00240
                                                    F60,F50);                F  00250
        CALL SQZBLNK (1, 1321);                                              F  00260
        CHAR1 = SUBSTR(STMT, 1, 1);                                          F  00270
        GO TO LBLTBL(INDEX(FIRSTCHR, CHAR1));                                F  00280
F10:    IF INDEX(STMT, '*16')=0                                              F  00290
          THEN CODE = '5';                                                   F  00300
          ELSE CODE = '6';                                                   F  00310
        STRT = 16;                                                           F  00320
        GO TO F70;                                                           F  00330
F20:    CODE = '2';                                                          F  00340
        STRT = 24;                                                           F  00350
        GO TO F70;                                                           F  00360
F30:    IF INDEX(STMT, '*2')=0                                               F  00370
          THEN CODE = '4';                                                   F  00380
          ELSE CODE = '3';                                                   F  00390
        STRT = 16;                                                           F  00400
        GO TO F70;                                                           F  00410
F40:    IF INDEX(STMT, '*1')=0                                               F  00420
          THEN CODE = '8';                                                   F  00430
          ELSE CODE = '7';                                                   F  00440
        STRT = 16;                                                           F  00450
        GO TO F70;                                                           F  00460
F50:    IF INDEX(STMT, '*8')=0                                               F  00470
          THEN CODE = '1';                                                   F  00480
```

```
            ELSE CODE = '2';                                                    F  00490
          STRT = 13;                                                            F  00500
          GO TO F70;                                                            F  00510
F60:      CALL NSTDST (1, NSTNO, CODE, STRT);                                   F  00520
          IF ERRFLG='1'B THEN DO;                                               F  00530
            ERRFLG = '0'B;                                                      F  00540
            RETURN;                                                             F  00550
            END;                                                                F  00560
          STRT = STRT + 8;                                                      F  00570
F70:      PAREN = INDEX(STMT, '(');                                             F  00580
          IF PAREN=0 THEN DO;                                                   F  00590
            CALL ERROR ('F -01: NO OPEN PAREN FOUND IN FUNCTION STMT.');        F  00600
            RETURN;                                                             F  00610
            END;                                                                F  00620
          STAR = INDEX(SUBSTR(STMT, STRT), '*') + STRT - 1;                     F  00630
          IF STAR=STRT-1 THEN STAR = 1321;                                      F  00640
          NXTDLM = MIN(STAR, PAREN);                                            F  00650
          IF NXTDLM-STRT>6 THEN DO;                                             F  00660
            CALL ERROR ('F -02: FUNC, IDENTIFIER HAS > 6 CHAR.');               F  00670
            RETURN;                                                             F  00680
            END;                                                                F  00690
          EXPLIST = EXPLIST || SUBSTR(STMT, STRT, NXTDLM-STRT) || '(' ||        F  00700
                    CODE || ')';                                                F  00710
          IF CHAR1¬='N' THEN DO;                                                F  00720
            CALL DUPEIT;                                                        F  00730
            RETURN;                                                             F  00740
            END;                                                                F  00750
          VARID = SUBSTR(STMT, STRT, NXTDLM-STRT);                              F  00760
          IF NXTDLM-STRT=6 THEN DO;                                             F  00770
            STMT = SUBSTR(STMT, 1, NXTDLM-2) || SUBSTR(STMT, NXTDLM);           F  00780
            NXTDLM = NXTDLM -1;                                                 F  00790
            END;                                                                F  00800
          EXPSTMT = PLUG(NSTNO) || VARID || '(' || NSDIM(NSTNO) || ')';         F  00810
          EXPNSL = EXPNSL || VARID || '|';                                      F  00820
          DIMLIST = DIMLIST || VARID || '|';                                    F  00830
          STMT = 'SUBROUTINE $' || SUBSTR(STMT, STRT, NXTDLM-STRT) ||           F  00840
                 ' (' || VARID || ',' || SUBSTR(STMT, NXTDLM+1);                F  00850
          CALL NUCARD;                                                          F  00860
F80:      RETURN;                                                               F  00870
          END FUNCST;                                                           F  00880
```

```
/*  I  (IMPSTMT)  28 JAN 70  THESIS VERSION  Do Ro LITTLE  */                I  00010
IMPSTMT:PROC;                                                                I  00020
        DCL ALPHBET       CHAR(27)          EXT,                           I  00030
            IMPCODE       CHAR(27)          EXT,                           I  00040
            IMPNSL        CHAR(27)VAR       EXT,                           I  00050
            NSDIM(6)      CHAR(3)VAR        EXT,                           I  00060
            NUMRLS        CHAR(10)          EXT,                           I  00070
            PLUG(8)       CHAR(11)VAR       EXT,                           I  00080
            STMT          CHAR(1320)VAR     EXT,                           I  00090
            CODE          CHAR(1),                                         I  00100
            NSTYPE        CHAR(1),                                         I  00110
            STRT1CHAR     CHAR(5)                 INIT('CILNR'),           I  00120
            SQZBLNK       ENTRY(FIXED,FIXED),                              I  00130
            LBLTBL(0:5)   LABEL                   INIT(I90, I20, I30,      I  00140
                                                       I40, I50, I60),      I  00150
            IMPSTOP       FIXED,                                           I  00160
            IMPSTRT       FIXED,                                           I  00170
            NSTNO         FIXED,                                           I  00180
            NXTCLPR       FIXED,                                           I  00190
            NXTCMMA       FIXED,                                           I  00200
            NXTDLM        FIXED,                                           I  00210
            SHORTEN       FIXED,                                           I  00220
            STMTEND       FIXED,                                           I  00230
            STRT          FIXED,                                           I  00240
            STRT1         FIXED;                                           I  00250
        CALL SQZBLNK (1, 1321);                                           I  00260
        STMTEND = LENGTH(STMT);                                           I  00270
        STRT1 = 9;                                                        I  00280
I10:    GO TO LBLTBL(INDEX(STRT1CHAR, SUBSTR(STMT, STRT1, 1)));           I  00290
I20:    IF SUBSTR(STMT, STRT1+7, 4)='*16(' THEN DO;                       I  00300
          CODE = '6';                                                     I  00310
          STRT = STRT1 + 11;                                              I  00320
          GO TO I70;                                                      I  00330
          END;                                                            I  00340
        IF SUBSTR(STMT, STRT1+7, 1)='(' THEN DO;                          I  00350
          CODE = '5';                                                     I  00360
          STRT = STRT1 + 8;                                               I  00370
          GO TO I70;                                                      I  00380
          END;                                                            I  00390
        IF SUBSTR(STMT, STRT1+7, 3)='*8(' THEN DO;                        I  00400
          CODE = '5';                                                     I  00410
          STRT = STRT1 + 10;                                              I  00420
          GO TO I70;                                                      I  00430
          END;                                                            I  00440
        GO TO I90;                                                        I  00450
I30:    IF SUBSTR(STMT,STRT1+7, 3)='*4(' THEN DO;                         I  00460
          CODE = '4';                                                     I  00470
          STRT = STRT1 + 10;                                              I  00480
```

```
            GO TO I70;                                                    I  00490
            END;                                                          I  00500
          IF SUBSTR(STMT,STRT1+7, 1)='(' THEN DO;                         I  00510
            CODE = '4';                                                   I  00520
            STRT = STRT1 + 8;                                             I  00530
            GO TO I70;                                                    I  00540
            END;                                                          I  00550
          IF SUBSTR(STMT,STRT1+7, 3)='*2(' THEN DO;                       I  00560
            CODE = '3';                                                   I  00570
            STRT = STRT1 + 10;                                            I  00580
            GO TO I70;                                                    I  00590
            END;                                                          I  00600
          GO TO I90;                                                      I  00610
I40:      IF SUBSTR(STMT, STRT1+7, 1)='(' THEN DO;                        I  00620
            CODE = '8';                                                   I  00630
            STRT = STRT1 + 8;                                             I  00640
            GO TO I70;                                                    I  00650
            END;                                                          I  00660
          IF SUBSTR(STMT, STRT1+7, 3)='*1(' THEN DO;                      I  00670
            CODE = '7';                                                   I  00680
            STRT = STRT1 + 10;                                            I  00690
            GO TO I70;                                                    I  00700
            END;                                                          I  00710
          IF SUBSTR(STMT, STRT1+7, 3)='*4(' THEN DO;                      I  00720
            CODE = '8';                                                   I  00730
            STRT = STRT1 + 10;                                            I  00740
            GO TO I70;                                                    I  00750
            END;                                                          I  00760
          GO TO I90;                                                      I  00770
I50:      CALL NSTDST (STRT1, NSTNO, CODE, STRT);                         I  00780
          IF ERRFLG='1'B THEN DO;                                         I  00790
            ERRFLG = '0'B;                                                I  00800
            RETURN;                                                       I  00810
            END;                                                          I  00820
          SHORTEN = STRT - STRT1 - LENGTH(PLUG(NSTNO));                   I  00830
          STMT = SUBSTR(STMT, 1, STRT1-1) || PLUG(NSTNO) ||               I  00840
                 SUBSTR(STMT, STRT);                                      I  00850
          STRT = STRT - SHORTEN + 1;                                      I  00860
          STMTEND = STMTEND - SHORTEN;                                    I  00870
          GO TO I70;                                                      I  00880
I60:      IF SUBSTR(STMT, STRT1+4, 1)='(' THEN DO;                        I  00890
            CODE = '1';                                                   I  00900
            STRT = STRT1 + 5;                                             I  00910
            GO TO I70;                                                    I  00920
            END;                                                          I  00930
          IF SUBSTR(STMT, STRT1+4, 3)='*8(' THEN DO;                      I  00940
            CODE = '2';                                                   I  00950
            STRT = STRT1 + 7;                                             I  00960
```

```
            GO TO I70;                                                          I  00970
            END;                                                                I  00980
          IF SUBSTR(STMT, STRT1+4, 3)='*4(' THEN DO;                            I  00990
            CODE = '1';                                                         I  01000
            STRT = STRT1 + 7;                                                   I  01010
            GO TO I70;                                                          I  01020
            END;                                                                I  01030
          GO TO I90;                                                            I  01040
I70:      NXTCLPR = INDEX(SUBSTR(STMT, STRT), ')') + STRT - 1;                  I  01050
          IF NXTCLPR=STRT-1 THEN NXTCLPR = STMTEND + 1;                         I  01060
          NXTCMMA = INDEX(SUBSTR(STMT, STRT), ',') + STRT - 1;                  I  01070
          IF NXTCMMA=STRT-1 THEN NXTCMMA = STMTEND + 1;                         I  01080
          NXTDLM = MIN (NXTCMMA, NXTCLPR);                                      I  01090
          IMPSTRT = INDEX(ALPHBET, SUBSTR(STMT, STRT, 1));                      I  01100
          IF IMPSTRT=0 THEN DO;                                                 I  01110
            CALL ERROR ('I -01: IMPLICIT SPEC. MUST DEAL W/A-Z,$.');            I  01120
            RETURN;                                                             I  01130
            END;                                                                I  01140
          IF NXTDLM-STRT=3 THEN DO;                                             I  01150
            IMPSTOP = INDEX(ALPHBET, SUBSTR(STMT, STRT+2, 1));                  I  01160
            IF IMPSTOP<IMPSTRT THEN DO;                                         I  01170
              CALL ERROR ('I -02: LAST CHAR IN IMP SPEC PRECEDES 1ST.');        I  01180
              RETURN;                                                           I  01190
              END;                                                              I  01200
            DO I=IMPSTRT TO IMPSTOP;                                            I  01210
              IMPCODE = SUBSTR(IMPCODE, 1, I-1) || CODE ||                      I  01220
                        SUBSTR(IMPCODE, I+1);                                   I  01230
              IF INDEX(NUMRLS, CODE)=0 THEN IMPNSL = IMPNSL ||                  I  01240
                 SUBSTR(ALPHBET, I, 1);                                         I  01250
              END;                                                              I  01260
            GO TO I80;                                                          I  01270
            END;                                                                I  01280
          IF NXTDLM-STRT=1 THEN DO;                                             I  01290
            IMPCODE = SUBSTR(IMPCODE, 1, IMPSTRT-1) || CODE ||                  I  01300
                      SUBSTR(IMPCODE, IMPSTRT+1);                               I  01310
            IF INDEX(NUMRLS, CODE)=0 THEN IMPNSL = IMPNSL ||                    I  01320
               SUBSTR(ALPHBET, IMPSTRT, 1);                                     I  01330
            GO TO I80;                                                          I  01340
            END;                                                                I  01350
          GO TO I90;                                                            I  01360
I80:      NXTDLM = MIN (NXTCMMA, NXTCLPR, STMTEND);                             I  01370
          IF NXTDLM=NXTCMMA THEN DO;                                            I  01380
            STRT = NXTCMMA + 1;                                                 I  01390
            GO TO I70;                                                          I  01400
            END;                                                                I  01410
          IF NXTCLPR=STMTEND THEN DO;                                           I  01420
            STMT = SUBSTR(STMT, 1, 8) || ' ' || SUBSTR(STMT, 9);                I  01430
            CALL NUCARD;                                                        I  01440
```

```
          RETURN;                                                          I  01450
          END;                                                             I  01460
        STRT1 = NXTCLPR + 2;                                               I  01470
        GO TO I10;                                                         I  01480
I90:    CALL ERROR ('I -03: UNRECOGNIZABLE STATEMEMT.');                   I  01490
        RETURN;                                                            I  01500
        END IMPSTMT;                                                       I  01510
```

```
/*  IF (IFSTMT)   01 AUG 70  THESIS VERSION  D. R. LITTLE  */                IF 00010
IFSTMT: PROC;                                                                IF 00020
        DCL EXPR          CHAR(1320)VAR  EXT,                                IF 00030
            LOCPNOS       CHAR(99)VAR    EXT,                                IF 00040
            RSLT          CHAR(30)VAR    EXT,                                IF 00050
            RSLTYP        CHAR(1)        EXT,                                IF 00060
            STMT          CHAR(1320)VAR  EXT,                                IF 00070
            STMTNO        CHAR(5)        EXT,                                IF 00080
            SUBSTMT       CHAR(1320)VAR,                                     IF 00090
            CONTNO        FIXED          EXT,                                IF 00100
            TYPN(8,2)     FIXED          EXT,                                IF 00110
            ERRFLG        BIT(1)         EXT,                                IF 00120
            LOCPEND       BIT(1)         EXT,                                IF 00130
            OLDSTMT       BIT(1)         EXT,                                IF 00140
            NSSTAT        BIT(1)         EXT,                                IF 00150
            NSEXPR        BIT(1),                                            IF 00160
            EQPOS         FIXED,                                             IF 00170
            ENDPR         FIXED,                                             IF 00180
            NXTCOMA       FIXED,                                             IF 00190
            NXTOPPR       FIXED,                                             IF 00200
            PARLOC        ENTRY(,FIXED,),                                    IF 00210
            SEARCH        ENTRY(CHAR(1320)VAR);                              IF 00220
        NSEXPR = '0'B;                                                       IF 00230
        IF OLDSTMT='1'B THEN DO;                                             IF 00240
          OLDSTMT = '0'B;                                                    IF 00250
          GO TO I10;                                                         IF 00260
          END;                                                               IF 00270
        CALL LITLOC;                                                         IF 00280
        IF ERRFLG='1'B THEN GO TO I50;                                       IF 00290
I10:    STMT = STMT || ';';                                                  IF 00300
        CALL PARLOC (STMT, 3, ENDPR);                                        IF 00310
        IF ERRFLG='1'B THEN GO TO I50;                                       IF 00320
        IF ENDPR=4 THEN DO;                                                  IF 00330
          CALL ERROR ('IF-01: OPEN AND CLOSE PARENS ADJACENT.');             IF 00340
          RETURN;                                                            IF 00350
          END;                                                               IF 00360
        STMT = SUBSTR(STMT, 1, LENGTH(STMT)-1);                              IF 00370
        EXPR = SUBSTR(STMT, 4, ENDPR-4);                                     IF 00380
        SUBSTMT = SUBSTR(STMT, ENDPR+1);                                     IF 00390
        NXTCOMA = INDEX(SUBSTMT, ',');                                       IF 00400
        IF NXTCOMA=0 | INDEX(SUBSTMT, '(')>0 THEN GO TO I20;                 IF 00410
        CALL SEARCH (EXPR);                                                  IF 00420
        IF NSSTAT='0'B THEN DO;                                              IF 00430
          CALL DUPEIT;                                                       IF 00440
          RETURN;                                                            IF 00450
          END;                                                               IF 00460
        CALL EXPRSN;                                                         IF 00470
        IF ERRFLG='1'B THEN GO TO I50;                                       IF 00480
```

```
            IF RSLTYP='7' | RSLTYP='8' THEN DO;                                   IF 00490
              CALL ERROR ('IF-02: ARITH IF STMT CONTAINS LOG EXPRSN.');           IF 00500
              RETURN;                                                             IF 00510
              END;                                                                IF 00520
            STMT = 'CALL N$7' || RSLTYP || '11 (T$7(1), ' || RSLT ||              IF 00530
                   ', 0.0)';                                                      IF 00540
            CALL NUCARD;                                                          IF 00550
            STMTNO = '     ';                                                     IF 00560
            STMT = 'IF (T$7(1)) GO TO ' || SUBSTR(SUBSTMT, 1, NXTCOMA-1);         IF 00570
            CALL NUCARD;                                                          IF 00580
            SUBSTMT = SUBSTR(SUBSTMT, NXTCOMA+1);                                 IF 00590
            NXTCOMA = INDEX(SUBSTMT, ',');                                        IF 00600
            IF NXTCOMA=0 THEN DO;                                                 IF 00610
              CALL ERROR ('IF-03: 2ND COMMA NOT FOUND IN ARITH IF STMT.');        IF 00620
              RETURN;                                                             IF 00630
              END;                                                                IF 00640
            IF TYPN(7, 2)=0 THEN TYPN(7, 2) = 1;                                  IF 00650
            STMT = 'CALL N$7' || RSLTYP || '31 (T$7(1), ' || RSLT ||              IF 00660
                   ', 0.0)';                                                      IF 00670
            CALL NUCARD;                                                          IF 00680
            STMT = 'IF (T$7(1)) GO TO ' || SUBSTR(SUBSTMT, 1, NXTCOMA-1);         IF 00690
            CALL NUCARD;                                                          IF 00700
            SUBSTMT = SUBSTR(SUBSTMT, NXTCOMA+1);                                 IF 00710
            STMT = 'GO TO ' || SUBSTMT;                                           IF 00720
            CALL NUCARD;                                                          IF 00730
            RETURN;                                                               IF 00740
I20:        CALL SEARCH (EXPR);                                                   IF 00750
            NSEXPR = NSSTAT;                                                      IF 00760
            NSSTAT = '0'B;                                                        IF 00770
            IF NSEXPR='0'B THEN GO TO I30;                                        IF 00780
            CALL EXPRSN;                                                          IF 00790
            IF ERRFLG='1'B THEN GO TO I50;                                        IF 00800
            EXPR = RSLT;                                                          IF 00810
I30:        EQPOS = INDEX(SUBSTMT, '=');                                          IF 00820
            NXTOPPR = INDEX(SUBSTMT, '(');                                        IF 00830
            IF NXTOPPR=0 THEN NXTOPPR = 1321;                                     IF 00840
            IF (EQPOS=0) | (NXTOPPR<EQPOS & INDEX(SUBSTMT, ')=')=0)               IF 00850
              THEN DO;                                                            IF 00860
                IF INDEX(SUBSTMT, 'IF')=1 | INDEX(SUBSTMT, 'CA')=1                IF 00870
                    THEN CALL SEARCH (':' || SUBSTMT);                            IF 00880
                IF INDEX(SUBSTMT, 'READ')=1 | INDEX(SUBSTMT, 'NA')=1 |            IF 00890
                   INDEX(SUBSTMT, 'DA')=1 THEN DO;                                IF 00900
                  STMT = 'IF (' || EXPR || ') ' || SUBSTMT;                       IF 00910
                  CALL NUCARD;                                                    IF 00920
                  GO TO I40;                                                      IF 00930
                  END;                                                            IF 00940
                END;                                                              IF 00950
              ELSE CALL SEARCH (SUBSTMT);                                         IF 00960
```

```
          IF NSSTAT='0'B THEN IF NSEXPR='0'B                                  IF CC970
            THEN DO;                                                          IF CC980
              CALL DUPEIT;                                                    IF CC990
              RETURN;                                                         IF 01000
              END;                                                            IF 01010
            ELSE DO;                                                          IF 01020
              STMT = 'IF (' || RSLT || ') ' || SUBSTMT;                       IF 01030
              CALL NUCARD;                                                    IF 01040
              RETURN;                                                         IF 01050
              END;                                                            IF 01060
          STMT = 'IF (.NOT.(' || EXPR || ')) GO TO ' ||                       IF 01070
                 SUBSTR(CONTNO, 4, 5);                                        IF 01080
          CALL NUCARD;                                                        IF 01090
          IF LENGTH(LOOPNOS)>93 THEN DO;                                      IF 01100
            CALL ERROR ('IF-04: DESIGN PARAMETER EXCEEDED: LOOPNOS.');        IF 01110
            RETURN;                                                           IF 01120
            END;                                                              IF 01130
          LOOPNOS = '.' || SUBSTR(CONTNO, 4, 5) || LOOPNOS;                   IF 01140
          CONTNO = CONTNO - 1;                                                IF 01150
          LOOPEND = '1'B;                                                     IF 01160
          STMTNO = '     ';                                                   IF 01170
I40:      STMT = SUBSTMT;                                                     IF 01180
          OLDSTMT = '1'B;                                                     IF 01190
          RETURN;                                                             IF 01200
I50:      ERRFLG = '0'B;                                                      IF 01210
          RETURN;                                                             IF 01220
          END IFSTMT;                                                         IF 01230
```

```
/*  L  (LITLOC)   28 JAN 70  THESIS VERSION  Do Ro LITTLE  */                    L  00010
LITLOC: PROC;                                                                    L  00020
        DCL  LITRLS        CHAR(1320)VAR  EXT,                                   L  00030
             NUMRLS        CHAR(10)       EXT,                                   L  00040
             STMT          CHAR(1320)VAR  EXT,                                   L  00050
             LITDLM        CHAR(4)                INIT('(/*,'),                  L  00060
             ERRFLG        BIT(1)         EXT,                                   L  00070
             DBLAPOS       FIXED,                                                L  00080
             LITLNGTH      FIXED,                                                L  00090
             K             FIXED,                                                L  00100
             NXTAPOS       FIXED,                                                L  00110
             NXTH          FIXED,                                                L  00120
             PNTR          FIXED,                                                L  00130
             PNTR2         FIXED,                                                L  00140
             SNGAPOS       FIXED,                                                L  00150
             SQZBLNK       ENTRY(FIXED,FIXED);                                   L  00160
        K = 1;                                                                   L  00170
        PNTR = 1;                                                                L  00180
        LITRLS = ';;';                                                           L  00190
L10:    NXTH = INDEX(SUBSTR(STMT, PNTR), 'H') + PNTR - 1;                        L  00200
        IF NXTH=PNTR-1 THEN NXTH = 1321;                                         L  00210
        NXTAPOS = INDEX(SUBSTR(STMT, PNTR), '''') + PNTR - 1;                    L  00220
        IF NXTAPOS=PNTR-1 THEN NXTAPOS = 1321;                                   L  00230
        IF NXTH<NXTAPOS THEN DO;                                                 L  00240
          CALL SQZBLNK (PNTR, NXTH);                                             L  00250
          GO TO L20;                                                             L  00260
          END;                                                                   L  00270
        IF NXTAPOS<NXTH THEN DO;                                                 L  00280
          CALL SQZBLNK (PNTR, NXTAPOS);                                          L  00290
          GO TO L40;                                                             L  00300
          END;                                                                   L  00310
        CALL SQZBLNK (PNTR, 1321);                                               L  00320
        RETURN;                                                                  L  00330
L20:    I = 2;                                                                   L  00340
L30:    IF INDEX(LITDLM, SUBSTR(STMT, NXTH-I, 1))=0 THEN DO;                     L  00350
          I = I + 1;                                                             L  00360
          IF I=5 THEN DO;                                                        L  00370
            PNTR = NXTH + 1;                                                     L  00380
            GO TO L10;                                                           L  00390
            END;                                                                 L  00400
          GO TO L30;                                                             L  00410
          END;                                                                   L  00420
        DO J=1 TO I-1;                                                           L  00430
          IF INDEX(NUMRLS, SUBSTR(STMT, NXTH-I+J, 1))=0 THEN DO;                 L  00440
            PNTR = NXTH + 1;                                                     L  00450
            GO TO L10;                                                           L  00460
            END;                                                                 L  00470
          END;                                                                   L  00480
```

```
          LITLNGTH = FIXED(SUBSTR(STMT, NXTH-I+1, I-1));                    L  00490
          IF NXTH+LITLNGTH>LENGTH(STMT) THEN DO;                            L  00500
            CALL ERROR ('L -01: STMT END PRECEDES LITERAL END.');           L  00510
            ERRFLG = '1'B;                                                  L  00520
            RETURN;                                                         L  00530
            END;                                                            L  00540
          LITRLS = ';;' || SUBSTR(K, 7, 2) || SUBSTR(STMT, NXTH+1,          L  00550
                   LITLNGTH) || LITRLS;                                     L  00560
          STMT = SUBSTR(STMT, 1, NXTH) || '''' || SUBSTR(K, 7, 2) ||        L  00570
                 '''' || SUBSTR(STMT, NXTH+LITLNGTH+1);                     L  00580
          K = K + 1;                                                        L  00590
          PNTR = NXTH + 5;                                                  L  00600
          GO TO L10;                                                        L  00610
L40:      PNTR2 = NXTAPOS + 1;                                              L  00620
L50:      DBLAPOS = INDEX(SUBSTR(STMT, PNTR2), '''''') + PNTR2 - 1;         L  00630
          IF DBLAPOS=PNTR2-1 THEN DBLAPOS = 1321;                           L  00640
          SNGAPOS = INDEX(SUBSTR(STMT, PNTR2), '''') + PNTR2 - 1;           L  00650
          IF SNGAPOS=PNTR2-1 THEN DO;                                       L  00660
            CALL ERROR ('L -02: NO CLOSE QUOTE FOUND.');                    L  00670
            ERRFLG = '1'B;                                                  L  00680
            RETURN;                                                         L  00690
            END;                                                            L  00700
          IF SNGAPOS<DBLAPOS THEN DO;                                       L  00710
            LITRLS = ';;' || SUBSTR(K, 7, 2) || SUBSTR(STMT, NXTAPOS,       L  00720
                     SNGAPOS-NXTAPOS+1) || LITRLS;                          L  00730
            STMT = SUBSTR(STMT, 1, NXTAPOS) || SUBSTR(K, 7, 2) ||           L  00740
                   SUBSTR(STMT, SNGAPOS);                                   L  00750
            K = K + 1;                                                      L  00760
            PNTR = NXTAPOS + 4;                                             L  00770
            GO TO L10;                                                      L  00780
            END;                                                            L  00790
          PNTR2 = DBLAPOS + 2;                                              L  00800
          GO TO L50;                                                        L  00810
          END LITLCC;                                                       L  00820
```

```
/*  N  (NUCARD)   01 FEB 70  THESIS VERSION  D. R. LITTLE  */                    N  00010
NUCARD: PROC;                                                                   N  00020
        DCL ACTFILE       CHAR(4)          EXT,                                 N  00030
            LITRLS        CHAR(1320)VAR    EXT,                                 N  00040
            OPSET(9)      CHAR(5)VAR       EXT,                                 N  00050
            STMT          CHAR(1320)VAR    EXT,                                 N  00060
            STMTNO        CHAR(5)          EXT,                                 N  00070
            LITSR         CHAR(4),                                              N  00080
            NEWCARD       CHAR(80),                                             N  00090
            OPRTR         CHAR(1),                                              N  00100
            I             FIXED,                                                N  00110
            LITLN         FIXED,                                                N  00120
            LITSTRT       FIXED,                                                N  00130
            LITSTOP       FIXED,                                                N  00140
            NXTLIT        FIXED,                                                N  00150
            NXTOP         FIXED,                                                N  00160
            SIZE          FIXED,                                                N  00170
            NSOUT         FILE OUTPUT      EXT,                                 N  00180
            NSWORK        FILE             EXT;                                 N  00190
        STRT = 1;                                                               N  00200
N10:    NXTOP = INDEX(SUBSTR(STMT, STRT), '>') + STRT - 1;                      N  00210
        IF NXTOP>STRT-1 THEN DO;                                                N  00220
          OPRTR = SUBSTR(STMT, NXTOP+1, 1);                                     N  00230
          IF OPRTR='N' THEN STMT = SUBSTR(STMT, 1, NXTOP-1) || '-' ||           N  00240
                                   SUBSTR(STMT, NXTOP+2);                       N  00250
          IF OPRTR='E' THEN STMT = SUBSTR(STMT, 1, NXTOP-1) || '**' ||          N  00260
                                   SUBSTR(STMT, NXTOP+2);                       N  00270
          IF INDEX('123456789', OPRTR)>0 THEN STMT = SUBSTR(STMT, 1,            N  00280
               NXTOP-1) || OPSET(FIXED(OPRTR)) || SUBSTR(STMT,NXTOP+2);         N  00290
          STRT = NXTOPR + 1;                                                    N  00300
          GO TO N10;                                                            N  00310
          END;                                                                  N  00320
        STRT = 1;                                                               N  00330
N20:    NXTLIT = INDEX(SUBSTR(STMT, STRT), '''') + STRT - 1;                    N  00340
        IF NXTLIT=STRT-1 THEN GO TO N30;                                        N  00350
        LITSR = ';;' || SUBSTR(STMT, NXTLIT+1, 2);                              N  00360
        LITSTRT = INDEX(LITRLS, LITSR);                                         N  00370
        LITSTOP = INDEX(SUBSTR(LITRLS, LITSTRT+1), ';;') + LITSTRT;             N  00380
        LITLN = LITSTOP - LITSTRT - 4;                                          N  00390
        STMT = SUBSTR(STMT, 1, NXTLIT-1) || SUBSTR(LITRLS, LITSTRT+4,           N  00400
               LITLN) || SUBSTR(STMT, NXTLIT+4);                                N  00410
        LITRLS = SUBSTR(LITRLS, 1, LITSTRT-1) || SUBSTR(LITRLS,                 N  00420
                 LITSTOP);                                                      N  00430
        STRT = NXTLIT + LITLN;                                                  N  00440
        GO TO N20;                                                              N  00450
N30:    SIZE = LENGTH(STMT);                                                    N  00460
        NEWCARD = STMTNO || ' ' || SUBSTR(STMT, 1, MIN(SIZE, 66));             N  00470
        IF ACTFILE='OUT '                                                       N  00480
```

```
    THEN PUT FILE (NSOUT) EDIT (NEWCARD) (A(80));                    N  00490
    ELSE PUT FILE (NSWORK) EDIT (NEWCARD) (A(80));                   N  00500
  IF SIZE<67 THEN RETURN;                                            N  00510
  DO I=2 TO SIZE/66 + 1;                                             N  00520
    NEWCARD = '     *' ||                                            N  00530
              SUBSTR(STMT, 66*I-65, MIN(SIZE-66*(I-1), 66));         N  00540
    IF ACTFILE='OUT '                                                N  00550
      THEN PUT FILE (NSOUT) EDIT (NEWCARD) (A(80));                  N  00560
      ELSE PUT FILE (NSWORK) EDIT (NEWCARD) (A(80));                 N  00570
    END;                                                             N  00580
  RETURN;                                                            N  00590
  END NUCARD;                                                        N  00600
```

```
/*  NA (NAMELST)  29 JAN 70  THESIS VERSION  Do Ro LITTLE  */                    NA 00010
NAMELST:PROC;                                                                   NA 00020
        DCL ALPHBET      CHAR(27)         EXT,                                  NA 00030
            EXPLIST      CHAR(2000)VAR    EXT,                                  NA 00040
            IMPCODE      CHAR(27)         EXT,                                  NA 00050
            IMPNSL       CHAR(27)VAR      EXT,                                  NA 00060
            STMT         CHAR(1320)VAR    EXT,                                  NA 00070
            CHAR1        CHAR(1),                                               NA 00080
            SYMB         CHAR(1),                                               NA 00090
            VARID        CHAR(6)VAR,                                            NA 00100
            ERRFLG       BIT(1)           EXT,                                  NA 00110
            OLDSTMT      BIT(1)           EXT,                                  NA 00120
            NSTNO        FIXED,                                                 NA 00130
            NXTDLM       FIXED,                                                 NA 00140
            STRT         FIXED,                                                 NA 00150
            SYMLOC       FIXED,                                                 NA 00160
            SQZBLNK      ENTRY(FIXED,FIXED);                                    NA 00170
        IF IMPNSL=(0)' ' THEN GO TO N40;                                        NA 00180
        CALL SQZBLNK (1, 1321);                                                 NA 00190
        STMT = STMT || ';';                                                     NA 00200
        NXTDLM = 9;                                                             NA 00210
N10:    STRT = INDEX(SUBSTR(STMT, NXTDLM+1), '/') + NXTDLM + 1;                 NA 00220
        IF STRT=NXTDLM+1 THEN DO;                                               NA 00230
          CALL ERROR ('NA-01: NO CLOSE SLASH FOUNDo');                          NA 00240
          RETURN;                                                               NA 00250
          END;                                                                  NA 00260
N20:    NXTDLM = 1321;                                                          NA 00270
        DO SYMB=',','/',';';                                                    NA 00280
          SYMLOC = INDEX(SUBSTR(STMT, STRT), SYMB);                             NA 00290
          IF SYMLOC =0                                                          NA 00300
            THEN SYMLOC = 1321;                                                 NA 00310
            ELSE SYMLOC = SYMLOC + STRT - 1;                                    NA 00320
          IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                                NA 00330
          END;                                                                  NA 00340
        IF NXTDLM-STRT>6 THEN DO;                                               NA 00350
          CALL ERROR ('NA-02: VAR IDENTIFIER HAS > 6 CHARo');                   NA 00360
          RETURN;                                                               NA 00370
          END;                                                                  NA 00380
        VARID = SUBSTR(STMT, STRT, NXTDLM-STRT);                                NA 00390
        CHAR1 = SUBSTR(STMT, STRT, 1);                                          NA 00400
        IF INDEX(ALPHBET, CHAR1)=0 THEN DO;                                     NA 00410
          CALL ERROR ('NA-03: ILLIGAL CHAR FOLLOWS COMMA OR SLASHo');           NA 00420
          RETURN;                                                               NA 00430
          END;                                                                  NA 00440
        IF INDEX(EXPLIST, ')'||VARID||'(')>0 THEN GO TO N30;                    NA 00450
        IF INDEX(IMPNSL, CHAR1)>0 THEN DO;                                      NA 00460
          NSTNO = INDEX(ALPHBET, SUBSTR(IMPCODE, INDEX(ALPHBET, CHAR1),         NA 00470
                  1));                                                          NA 00480
```

```
            CALL DIMENIT (VARID, NSTNO);                                NA 00490
            IF ERRFLG='1'B THEN DO;                                     NA 00500
              ERRFLG = '0'B;                                            NA 00510
              RETURN;                                                   NA 00520
              END;                                                      NA 00530
            END;                                                        NA 00540
N30:        SYMB = SUBSTR(STMT, NXTDLM, 1);                             NA 00550
            IF SYMB=',' THEN DO;                                        NA 00560
              STRT = NXTDLM + 1;                                        NA 00570
              GO TO N20;                                                NA 00580
              END;                                                      NA 00590
            IF SYMB='/' THEN GO TO N10;                                 NA 00600
N40:        IF OLDSTMT='1'B                                             NA 00610
              THEN OLDSTMT = '0'B;                                      NA 00620
              ELSE CALL DUPEIT;                                         NA 00630
            RETURN;                                                     NA 00640
            END NAMELST;                                                NA 00650
```

```
/*  NS (NSTDST)   29 JAN 70  THESIS VERSION  D. R. LITTLE  */                    NS 00010
NSTDST: PROC (INIT, NSTNO, CODE, STRT);                                          NS 00020
        DCL  ALPHBET       CHAR(27)          EXT,                              NS 00030
             NSDIM(6)      CHAR(3)VAR        EXT,                              NS 00040
             NUMRLS        CHAR(10)          EXT,                              NS 00050
             STMT          CHAR(1320)VAR     EXT,                              NS 00060
             CODE          CHAR(1),                                            NS 00070
             NSTDDIM       CHAR(3)VAR,                                         NS 00080
             NSTYPE        CHAR(1),                                            NS 00090
             ERRFLG        BIT(1)            EXT,                              NS 00100
             INIT          FIXED,                                              NS 00110
             NSTNO         FIXED,                                              NS 00120
             STAR          FIXED,                                              NS 00130
             STRT          FIXED;                                              NS 00140
        STAR = INDEX(SUBSTR(STMT, INIT), '*');                                   NS 00150
        IF STAR¬=6 & STAR¬=8 & STAR¬=13 THEN DO;                                  NS 00160
          CALL ERROR ('NS-01: NONSTANDARD SYNTAX ERROR.');                       NS 00170
          GO TO N20;                                                             NS 00180
          END;                                                                   NS 00190
        STAR = STAR + INIT - 1;                                                  NS 00200
        NSTYPE = SUBSTR(STMT, STAR-1, 1);                                        NS 00210
        IF INDEX('123456', NSTYPE)=0 THEN DO;                                    NS 00220
          CALL ERROR ('NS-02: NONSTANDARD TYPE IS NOT 1-6.');                    NS 00230
          GO TO N20;                                                             NS 00240
          END;                                                                   NS 00250
        IF INDEX(NUMRLS, SUBSTR(STMT, STAR+1, 1))=0 THEN DO;                     NS 00260
          CALL ERROR ('NS-03: NONSTANDARD SIZE IS UNRECOGNIZABLE.');             NS 00270
          GO TO N20;                                                             NS 00280
          END;                                                                   NS 00290
        I = 1;                                                                   NS 00300
N10:    IF INDEX(NUMRLS, SUBSTR(STMT, STAR+1+I, 1))>0 THEN DO;                   NS 00310
          I = I + 1;                                                             NS 00320
          IF I=4 THEN DO;                                                        NS 00330
            CALL ERROR ('NS-04: NONSTD SIZE EXCEEDS THREE DIGITS.');             NS 00340
            GO TO N20;                                                           NS 00350
            END;                                                                 NS 00360
          GO TO N10;                                                             NS 00370
          END;                                                                   NS 00380
        NSTDDIM = SUBSTR(STMT, STAR+1, I);                                       NS 00390
        NSTNO = NSTYPE;                                                          NS 00400
        IF NSDIM(NSTNO)¬=0 & NSDIM(NSTNO)¬=NSTDDIM THEN DO;                      NS 00410
          CALL ERROR ('NS-05: NONSTD-N USED W/DIFFERENT SIZE.');                 NS 00420
          GO TO N20;                                                             NS 00430
          END;                                                                   NS 00440
        NSDIM(NSTNO) = NSTDDIM;                                                  NS 00450
        CODE = SUBSTR(ALPHBET, NSTNO, 1);                                        NS 00460
        STRT = STAR + 1 + I;                                                     NS 00470
        RETURN;                                                                  NS 00480
```

```
N20:    ERRFLG = '1'B;                                          NS 00490
        RETURN;                                                 NS 00500
        END NSTDST;                                             NS 00510
```

```
/*  P  (PROCCRD)  17 MAR 70  THESIS VERSION  Do Ro LITTLE  */                    P  00010
PROCCRD:PROC;                                                                    P  00020
        DCL ACTFILE       CHAR(4)          EXT,                                P  00030
            CARD(21)      CHAR(80)         EXT,                                P  00040
            LOOPNOS       CHAR(99)VAR      EXT,                                P  00050
            STMT          CHAR(1320)VAR    EXT,                                P  00060
            STMTNO        CHAR(5)          EXT,                                P  00070
            CHAR1         CHAR(1),                                             P  00080
            CHAR2         CHAR(1),                                             P  00090
            FIRSTCHAR     CHAR(15)              INIT('ABCDEFGILNPRSTW'),       P  00100
            SECNDCHAR     CHAR(14)              INIT('AEFILMNOQRSTUX'),        P  00110
            STNO          CHAR(5)VAR,                                          P  00120
            SYMB          CHAR(1),                                             P  00130
            LOOPEND       BIT(1)           EXT,                                P  00140
            OLDSTMT       BIT(1)           EXT,                                P  00150
            CARDNO        FIXED            EXT,                                P  00160
            FUNC          FIXED,                                               P  00170
            NXTBLNK       FIXED,                                               P  00180
            NXTDLM        FIXED,                                               P  00190
            SYMLOC        FIXED,                                               P  00200
            SQZBLNK       ENTRY(FIXED,FIXED),                                  P  00210
            LBLTBL(0:15,0:14) LABEL             INIT((26)P110,(2)P120,         P  00220
                 (3)P110,P120,(3)P110,P120,(10)P110,P130,(6)P110,P60,            P  00230
                 (7)P110,P150,P120,P110,P70,(3)P110,P80,(13)P110,P90,            P  00240
                 P110,P190,(4)P110,P120,(4)P110,P120,(3)P110,P120,               P  00250
                 (4)P110,P120,(9)P110,P120,(9)P110,P160,(2)P110,P180,            P  00260
                 P140,(15)P110,P140,(7)P110,P170,(6)P110,P140,(2)P110,           P  00270
                 P140,(4)P110,P120,(8)P110,P120,(2)P110,P120,(3)P110,            P  00280
                 P100,(24)P110,(2)P120,(11)P110,P120,(14)P110,P120,              P  00290
                 (4)P110);                                                       P  00300
        OLDSTMT = '0'B;                                                          P  00310
        STMTNO = SUBSTR (CARD(1), 1, 5);                                         P  00320
        IF STMTNO='     ' | LENGTH(LOOPNOS)=0 THEN GO TO P30;                    P  00330
        I = 1;                                                                   P  00340
P10:    IF SUBSTR(STMTNO, I, 1)=' ' | SUBSTR(STMTNO, I, 1)='0' THEN DO;         P  00350
          I = I + 1;                                                             P  00360
          GO TO P10;                                                             P  00370
          END;                                                                   P  00380
        STRT = I;                                                                P  00390
        I = 0;                                                                   P  00400
P20:    IF SUBSTR(STMTNO, 5-I, 1)=' ' THEN DO;                                   P  00410
          I = I + 1;                                                             P  00420
          GO TO P20;                                                             P  00430
          END;                                                                   P  00440
        STNO = SUBSTR(STMTNO, STRT, 6-I-STRT);                                   P  00450
        IF STNO||'.'=SUBSTR(LOOPNOS, 1, 7-I-STRT) THEN LOOPEND = '1'B;           P  00460
P30:    STMT = SUBSTR (' ', 1, 0);                                               P  00470
        DO I=1 TO CARDNO;                                                        P  00480
```

```
            STMT = STMT || SUBSTR (CARD(I), 7, 66);                          P  00490
            END;                                                             P  00500
          NXTDLM = 1321;                                                     P  00510
          DO SYMB='(','/','=';                                               P  00520
            SYMLOC = INDEX(STMT, SYMB);                                      P  00530
            IF SYMLOC=0 THEN SYMLOC = 1321;                                  P  00540
            IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                           P  00550
            END;                                                             P  00560
          CALL SQZBLNK (1, NXTDLM);                                          P  00570
          IF ACTFILE='OUT ' THEN DO;                                         P  00580
            IF INDEX(STMT, 'IMPLICIT')=1 THEN GO TO P40;                     P  00590
            IF INDEX(STMT, 'SUBROUTINE')=1 THEN GO TO P40;                   P  00600
            FUNC = INDEX(STMT, 'FUNCTION');                                  P  00610
            IF FUNC>0 & FUNC<18 THEN GO TO P40;                              P  00620
            ACTFILE = 'WORK';                                                P  00630
            END;                                                             P  00640
P40:      CHAR1 = SUBSTR (STMT, 1, 1);                                       P  00650
          CHAR2 = SUBSTR (STMT, 2, 1);                                       P  00660
          IF NXTDLM>7 | INDEX(STMT, '=')=0 | INDEX(STMT, ',')¬=0 &           P  00670
              INDEX(STMT, '(')=0 THEN GO TO P50;                             P  00680
          IF INDEX(STMT, 'CALL')=1 | INDEX(STMT, 'DATA')=1 |                 P  00690
              INDEX(STMT, 'ENTRY')=1 | INDEX(STMT, 'FORMAT(')=1 |            P  00700
              INDEX(STMT, 'IF(')=1 | INDEX(STMT, 'REAL')=1                   P  00710
              THEN GO TO P50;                                                P  00720
          IF INDEX(STMT, 'READ(')=1 THEN GO TO P50;                          P  00730
          IF INDEX(STMT, 'WRITE(')=1 THEN GO TO P50;                         P  00740
          CALL ASGMTST;                                                      P  00750
          RETURN;                                                            P  00760
P50:      GO TO LBLTBL(INDEX(FIRSTCHAR, CHAR1), INDEX(SECNDCHAR, CHAR2));    P  00770
P60:      IF SUBSTR(STMT, 4, 1)='M' THEN DO;                                 P  00780
            CALL CCMSTMT;                                                    P  00790
            RETURN;                                                          P  00800
            END;                                                             P  00810
          IF SUBSTR(STMT, 4, 1)='P' THEN GO TO P140;                         P  00820
          CALL DUPEIT;                                                       P  00830
          RETURN;                                                            P  00840
P70:      IF SUBSTR(STMT, 3, 1)='M'                                          P  00850
            THEN CALL DIMENST;                                               P  00860
            ELSE CALL DUPEIT;                                                P  00870
          RETURN;                                                            P  00880
P80:      IF INDEX(STMT, '=')>0 THEN DO;                                     P  00890
            CALL DOSTMT;                                                     P  00900
            RETURN;                                                          P  00910
            END;                                                             P  00920
          IF INDEX(STMT, 'DOUBLEPRECISION')=1 THEN GO TO P140;               P  00930
          CALL DUPEIT;                                                       P  00940
          RETURN;                                                            P  00950
P90:      IF LENGTH(STMT)=3                                                  P  00960
```

```
          THEN CALL ENDSTMT;                                                    P  00970
          ELSE CALL DUPEIT;                                                     P  00980
        RETURN;                                                                 P  00990
P100:   IF SUBSTR(STMT, 4, 1)='D' THEN DO;                                      P  01000
          CALL READST;                                                          P  01010
          RETURN;                                                               P  01020
          END;                                                                  P  01030
        IF SUBSTR(STMT, 4, 1)='L' THEN GO TO P140;                              P  01040
        CALL DUPEIT;                                                            P  01050
        RETURN;                                                                 P  01060
P110:   CALL ERRCR ('P -01: UNRECOGNIZABLE STATEMENT.');                        P  01070
        RETURN;                                                                 P  01080
P120:   CALL DUPEIT;                                                            P  01090
        RETURN;                                                                 P  01100
P130:   CALL CALLST;                                                            P  01110
        RETURN;                                                                 P  01120
P140:   FUNC = INDEX(STMT, 'FUNCTION');                                         P  01130
        IF FUNC>4 & FUNC<18                                                     P  01140
          THEN CALL FUNCST;                                                     P  01150
          ELSE CALL TYPSTMT;                                                    P  01160
        RETURN;                                                                 P  01170
P150:   CALL DATAST;                                                            P  01180
        RETURN;                                                                 P  01190
P160:   CALL IFSTMT;                                                            P  01200
        IF OLDSTMT='1'B THEN GO TO P40;                                         P  01210
        RETURN;                                                                 P  01220
P170:   CALL NAMELST;                                                           P  01230
        RETURN;                                                                 P  01240
P180:   CALL IMPSTMT;                                                           P  01250
        RETURN;                                                                 P  01260
P190:   CALL EQUIVST;                                                           P  01270
        RETURN;                                                                 P  01280
    1   END PROCCRD;                                                            P  01290
```

```
/*  PA (PARLOC)   29 JAN 70  THESIS VERSION  D. R. LITTLE  */                PA 00010
PARLOC: PROC (STMTPR, STRTPR, ENDPR);                                        PA 00020
        DCL STMTPR          CHAR(1320)VAR,                                   PA 00030
            SYMB            CHAR(1),                                         PA 00040
            ERRFLG          BIT(1)          EXT,                             PA 00050
            ENDPR           FIXED,                                           PA 00060
            NCLPR           FIXED,                                           PA 00070
            NOPPR           FIXED,                                           PA 00080
            STRT            FIXED,                                           PA 00090
            STRTPR          FIXED,                                           PA 00100
            SYMLOC          FIXED;                                           PA 00110
        STRT = STRTPR + 1;                                                   PA 00120
        NOPPR = 1;                                                           PA 00130
        NCLPR = 0;                                                           PA 00140
P10:    ENDPR = 1321;                                                        PA 00150
        DO SYMB='(',')',';';                                                 PA 00160
          SYMLOC = INDEX(SUBSTR(STMTPR, STRT), SYMB);                        PA 00170
          IF SYMLOC=0                                                        PA 00180
            THEN SYMLOC = 1321;                                              PA 00190
            ELSE SYMLOC = SYMLOC + STRT - 1;                                 PA 00200
          IF SYMLOC<ENDPR THEN ENDPR = SYMLOC;                               PA 00210
          END;                                                               PA 00220
        SYMB = SUBSTR(STMTPR, ENDPR, 1);                                     PA 00230
        IF SYMB=';' THEN DO;                                                 PA 00240
          CALL ERROR ('PA-01: UNMATCHED PARENS FOUND.');                     PA 00250
          ERRFLG = '1'B;                                                     PA 00260
          RETURN;                                                            PA 00270
          END;                                                               PA 00280
        IF SYMB='('                                                          PA 00290
          THEN NCPPR = NOPPR + 1;                                            PA 00300
          ELSE NCLPR = NCLPR + 1;                                            PA 00310
        IF NCLPR<NOPPR THEN DO;                                              PA 00320
          STRT = ENDPR + 1;                                                  PA 00330
          GO TO P10;                                                         PA 00340
          END;                                                               PA 00350
        RETURN;                                                              PA 00360
        END PARLOC;                                                          PA 00370
```

```
/*  R  (READST)   04 SEP 70  THESIS VERSION  D. R. LITTLE  */                R  00010
READST: PROC;                                                                R  00020
        DCL  ALPHBET       CHAR(27)          EXT,                            R  00030
             EXPLIST       CHAR(2000)VAR     EXT,                            R  00040
             IMPCODE       CHAR(27)          EXT,                            R  00050
             IMPNSL        CHAR(27)VAR       EXT,                            R  00060
             STMT          CHAR(1320)VAR     EXT,                            R  00070
             VARID         CHAR(6)VAR        EXT,                            R  00080
             CHAR1         CHAR(1),                                          R  00090
             SYMB          CHAR(1),                                          R  00100
             ERRFLG        BIT(1)            EXT,                            R  00110
             NSSTAT        BIT(1)            EXT,                            R  00120
             OLDSTMT       BIT(1)            EXT,                            R  00130
             ENDPR         FIXED,                                            R  00140
             NSTNO         FIXED,                                            R  00150
             NXTDLM        FIXED,                                            R  00160
             STRT          FIXED,                                            R  00170
             SYMLOC        FIXED,                                            R  00180
             SEARCH        ENTRY(CHAR(1320)VAR),                             R  00190
             SQZBLNK       ENTRY(FIXED,FIXED);                               R  00200
        CALL SQZBLNK (1,1321);                                               R  00210
        IF IMPNSL=(0)' ' THEN GO TO R40;                                     R  00220
        IF INDEX(STMT, ',')=0 & INDEX(STMT, '(')=0 THEN GO TO R40;           R  00230
        IF SUBSTR(STMT, 5, 1)='('                                            R  00240
          THEN STRT = INDEX(STMT, ')') + 1;                                  R  00250
          ELSE STRT = INDEX(STMT, ',') + 1;                                  R  00260
        IF STRT=1 THEN DO;                                                   R  00270
          CALL ERROR ('R -01: UNFAMILIAR READ STMT SYNTAX.');                R  00280
          RETURN;                                                            R  00290
          END;                                                               R  00300
        STMT = STMT || ';';                                                  R  00310
        CALL SEARCH (SUBSTR(STMT, STRT));                                    R  00320
        IF NSSTAT='0'B THEN GO TO R40;                                       R  00330
R10:    CHAR1 = SUBSTR(STMT, STRT, 1);                                       R  00340
        IF CHAR1='(' THEN DO;                                                R  00350
          CALL PARLOC (STMT, STRT, ENDPR);                                   R  00360
          IF ERRFLG='1'B THEN GO TO R50;                                     R  00370
          NXTDLM = ENDPR + 1;                                                R  00380
          SYMB = SUBSTR(STMT, NXTDLM, 1);                                    R  00390
          GO TO R30;                                                         R  00400
          END;                                                               R  00410
        IF INDEX(ALPHBET, CHAR1)=0 THEN DO;                                  R  00420
          CALL ERROR ('R -02: ILLEGAL 1ST CHAR IN I/O LIST ITEM.');          R  00430
          RETURN;                                                            R  00440
          END;                                                               R  00450
        NXTDLM = 1321;                                                       R  00460
        DO SYMB=')',',',';';                                                 R  00470
          SYMLOC = INDEX(SUBSTR(STMT, STRT), SYMB);                          R  00480
```

```
            IF SYMLOC=0                                                          R  00490
              THEN SYMLOC = 1321;                                                R  00500
              ELSE SYMLOC = SYMLOC + STRT -1;                                    R  00510
            IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                               R  00520
            END;                                                                 R  00530
          IF NXTDLM-STRT>6 THEN DO;                                              R  00540
            CALL ERROR ('R -03: VAR IDENTIFIER HAS > 6 CHAR.');                  R  00550
            RETURN;                                                              R  00560
            END;                                                                 R  00570
          SYMB = SUBSTR(STMT, NXTDLM, 1);                                        R  00580
          VARID = SUBSTR(STMT, STRT, NXTDLM-STRT);                               R  00590
          IF INDEX(IMPNSL, SUBSTR(VARID, 1, 1))>0 THEN DO;                       R  00600
            IF INDEX(EXPLIST, ')'||VARID||'(')>0 THEN GO TO R20;                 R  00610
            NSTNO = INDEX(ALPHBET, SUBSTR(IMPCODE, INDEX(ALPHBET, CHAR1),       R  00620
                    1));                                                         R  00630
            CALL DIMENIT (VARID, NSTNO);                                         R  00640
            IF ERRFLG='1'B THEN GO TO R50;                                       R  00650
            END;                                                                 R  00660
R20:      IF SYMB='(' THEN DO;                                                   R  00670
            CALL PARLOC (STMT, NXTDLM, ENDPR);                                   R  00680
            IF ERRFLG='1'B THEN GO TO R50;                                       R  00690
            NXTDLM = ENDPR + 1;                                                  R  00700
            SYMB = SUBSTR(STMT, NXTDLM, 1);                                      R  00710
            END;                                                                 R  00720
R30:      IF SYMB=',' THEN DO;                                                   R  00730
            STRT = NXTDLM + 1;                                                   R  00740
            GO TO R10;                                                           R  00750
            END;                                                                 R  00760
          IF SYMB=';' THEN DO;                                                   R  00770
R40:        IF OLDSTMT='1'B                                                      R  00780
              THEN OLDSTMT = '0'B;                                               R  00790
              ELSE CALL DUPEIT;                                                  R  00800
            RETURN;                                                              R  00810
            END;                                                                 R  00820
          CALL ERROR ('R -04: READ STMT SYNTAX ERROR.');                         R  00830
          RETURN;                                                                R  00840
R50:      ERRFLG = '0'B;                                                         R  00850
          RETURN;                                                                R  00860
          END READST;                                                            R  00870
```

```
/*  S  (SQZBLNK)  28 JAN 70  THESIS VERSION  Do Ro LITTLE  */                    S  00010
SQZBLNK:PROC (STRTSQZ, STOPSQZ);                                                 S  00020
        DCL STMT          CHAR(1320)VAR  EXT,                                    S  00030
            NXTBLNK       FIXED,                                                 S  00040
            STOPSQZ       FIXED,                                                 S  00050
            STRTSQZ       FIXED;                                                 S  00060
S10:    NXTBLNK = INDEX (SUBSTR(STMT, STRTSQZ), ' ') + STRTSQZ - 1;              S  00070
        IF NXTBLNK=STRTSQZ-1 | NXTBLNK>STOPSQZ THEN RETURN;                      S  00080
        STMT = SUBSTR(STMT, 1, NXTBLNK-1) || SUBSTR(STMT, NXTBLNK+1);            S  00090
        STOPSQZ = STOPSQZ - 1;                                                   S  00100
        GO TO S10;                                                               S  00110
        END SQZBLNK;                                                             S  00120
```

```
/*  SR (SEARCH)   29 JAN 70  THESIS VERSION  D. R. LITTLE  */                SR 00010
SEARCH: PROC (STMTSR);                                                       SR 00020
        DCL EXPLIST      CHAR(1000)VAR  EXT,                                 SR 00030
            EXPNSL       CHAR(2000)VAR  EXT,                                 SR 00040
            IMPNSL       CHAR(27)VAR    EXT,                                 SR 00050
            STMTSR       CHAR(1320)VAR,                                      SR 00060
            LEFTDLM      CHAR(8)              INIT('=+-*/(,. '),             SR 00070
            RIGHTDLM     CHAR(11)             INIT('=+-*/(),>.;'),           SR 00080
            SYMB         CHAR(1),                                            SR 00090
            NSSTAT       BIT(1)         EXT,                                 SR 00100
            BAR          FIXED,                                              SR 00110
            IMPNSLT      FIXED,                                              SR 00120
            LEFT         FIXED,                                              SR 00130
            NSLEND       FIXED,                                              SR 00140
            NSLOC        FIXED,                                              SR 00150
            NSLOCT       FIXED,                                              SR 00160
            NXTBAR       FIXED,                                              SR 00170
            NXTDLM       FIXED,                                              SR 00180
            RIGHT        FIXED,                                              SR 00190
            STRT         FIXED,                                              SR 00200
            STSREND      FIXED,                                              SR 00210
            SYMLOC       FIXED;                                              SR 00220
        NSSTAT = '0'B;                                                       SR 00230
        NSLEND = LENGTH(EXPNSL);                                             SR 00240
        STSREND = LENGTH(STMTSR);                                            SR 00250
        IF NSLEND=1 THEN GO TO S40;                                          SR 00260
        BAR = 1;                                                             SR 00270
S10:    STRT = 1;                                                            SR 00280
S20:    NXTBAR = INDEX(SUBSTR(EXPNSL, BAR+1), '|') + BAR;                    SR 00290
        NSLOC = INDEX(SUBSTR(STMTSR, STRT), SUBSTR(EXPNSL, BAR+1,            SR 00300
                NXTBAR-BAR-1)) + STRT - 1;                                   SR 00310
        IF NSLOC=STRT-1 THEN DO;                                             SR 00320
S30:      IF NXTBAR=NSLEND THEN GO TO S40;                                   SR 00330
          BAR = NXTBAR;                                                      SR 00340
          GO TO S10;                                                         SR 00350
          END;                                                               SR 00360
        IF NSLOC=1 | SUBSTR(STMTSR, NSLOC-2, 1)='>'                          SR 00370
          THEN LEFT = 9;                                                     SR 00380
          ELSE LEFT = INDEX(LEFTDLM, SUBSTR(STMTSR, NSLOC-1, 1));            SR 00390
        NSLOCT = NSLOC + NXTBAR - BAR - 2;                                   SR 00400
        IF NSLOCT=STSREND                                                    SR 00410
          THEN RIGHT = 12;                                                   SR 00420
          ELSE RIGHT = INDEX(RIGHTDLM, SUBSTR(STMTSR, NSLOCT+1, 1));        SR 00430
        IF LEFT¬=0 & RIGHT¬=0 THEN DO;                                       SR 00440
          NSSTAT = '1'B;                                                     SR 00450
          RETURN;                                                            SR 00460
          END;                                                               SR 00470
        STRT = NSLOC + 1;                                                    SR 00480
```

```
          IF STRT=STSREND+1 THEN GO TO S30;                                    SR 00490
          GO TO S20;                                                           SR 00500
S40:      IMPNSLT = LENGTH(IMPNSL);                                            SR 00510
          IF IMPNSLT=0 THEN RETURN;                                            SR 00520
          I = 1;                                                               SR 00530
S50:      STRT = 1;                                                            SR 00540
S60:      NSLOC = INDEX(SUBSTR(STMTSR, STRT), SUBSTR(IMPNSL, I, 1)) +          SR 00550
                STRT - 1;                                                      SR 00560
          IF NSLOC=STRT-1 THEN DO;                                             SR 00570
S70:        IF I=IMPNSLT THEN RETURN;                                          SR 00580
            I = I + 1;                                                         SR 00590
            GO TO S50;                                                         SR 00600
            END;                                                               SR 00610
          IF NSLOC=1 | SUBSTR(STMTSR, NSLOC-2, 1)='>'                          SR 00620
            THEN LEFT = 9;                                                     SR 00630
            ELSE LEFT = INDEX(LEFTDLM, SUBSTR(STMTSR, NSLOC-1, 1));          SR 00640
          IF LEFT¬=0 THEN DO;                                                  SR 00650
            NXTDLM = STSREND + 1;                                              SR 00660
            DO SYMB='=','+','-','*','/','(',')',',','>','°',';';              SR 00670
              SYMLOC = INDEX(SUBSTR(STMTSR, NSLOC), SYMB);                     SR 00680
              IF SYMLOC=0                                                      SR 00690
                THEN SYMLOC = 1321;                                            SR 00700
                ELSE SYMLOC = SYMLOC + NSLOC - 1;                              SR 00710
              IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                           SR 00720
              END;                                                             SR 00730
            IF INDEX(EXPLIST, ')'||SUBSTR(STMTSR, NSLOC, NXTDLM-NSLOC)        SR 00740
                ||'(')>0 THEN DO;                                              SR 00750
              STRT = NXTDLM;                                                   SR 00760
              GO TO S80;                                                       SR 00770
              END;                                                             SR 00780
            IF NXTDLM<STSREND & NSLOC>1 THEN IF INDEX(                         SR 00790
                '°LT°LE°EQ°NE°GE°GT°AND°OR°NOT°,END=,ERR=', SUBSTR(           SR 00800
                STMTSR, NSLOC-1, NXTDLM-NSLOC+2))>0 THEN DO;                   SR 00810
              STRT = NXTDLM;                                                   SR 00820
              GO TO S80;                                                       SR 00830
              END;                                                             SR 00840
            NSSTAT = '1'B;                                                     SR 00850
            RETURN;                                                            SR 00860
            END;                                                               SR 00870
          STRT = NSLOC + 1;                                                    SR 00880
S80:      IF STRT=STSREND+1 THEN GO TO S70;                                    SR 00890
          GO TO S60;                                                           SR 00900
          END SEARCH;                                                          SR 00910
```

```
/*  T  (TYPSTMT)  17 MAR 70  THESIS VERSION  Do Ro LITTLE  */                    T  00010
TYPSTMT:PROC;                                                                    T  00020
        DCL ALPHBET       CHAR(27)          EXT,                               T  00030
            DIMLIST       CHAR(1000)VAR     EXT,                               T  00040
            EXPLIST       CHAR(2000)VAR     EXT,                               T  00050
            EXPNSL        CHAR(1000)VAR     EXT,                               T  00060
            NSDIM(6)      CHAR(3)VAR        EXT,                               T  00070
            NSFNS         CHAR(141)VAR      EXT,                               T  00080
            NUMRLS        CHAR(10)          EXT,                               T  00090
            PLUG(8)       CHAR(11)VAR       EXT,                               T  00100
            STMT          CHAR(1320)VAR     EXT,                               T  00110
            CHAR1         CHAR(1),                                             T  00120
            CODE          CHAR(1),                                             T  00130
            FIRSTCHAR     CHAR(6)                 INIT('CDILNR'),              T  00140
            NSTYPE        CHAR(1),                                             T  00150
            SYMB          CHAR(1),                                             T  00160
            TCODE         CHAR(1),                                             T  00170
            ERRFLG        BIT(1)            EXT,                               T  00180
            CLPR          FIXED,                                               T  00190
            CLSLSH        FIXED,                                               T  00200
            NSTNC         FIXED,                                               T  00210
            NXTDLM        FIXED,                                               T  00220
            STRT          FIXED,                                               T  00230
            SYMLOC        FIXED,                                               T  00240
            NSTDST        ENTRY(FIXED,,,),                                     T  00250
            LBLTBL(0:6)   LABEL                   INIT(T80,T10,T20,T30,T40,    T  00260
                                                  T50,T60);                    T  00270
        CALL LITLOC;                                                         T  00280
        IF ERRFLG='1'B THEN DO;                                              T  00290
          ERRFLG = '0'B;                                                     T  00300
          RETURN;                                                            T  00310
          END;                                                               T  00320
        STMT = STMT || ';';                                                  T  00330
        CHAR1 = SUBSTR(STMT, 1, 1);                                          T  00340
        GO TO LBLTBL(INDEX(FIRSTCHAR, CHAR1));                               T  00350
T10:    IF SUBSTR(STMT, 8, 3)='*16' THEN DO;                                 T  00360
          CODE = '6';                                                        T  00370
          STRT = 11;                                                         T  00380
          GO TO T70;                                                         T  00390
          END;                                                               T  00400
        CODE = '5';                                                          T  00410
        IF SUBSTR(STMT, 8, 2)='*8'                                           T  00420
          THEN STRT = 10;                                                    T  00430
          ELSE STRT = 8;                                                     T  00440
        GO TO T70;                                                           T  00450
T20:    CODE = '2';                                                          T  00460
        STRT = 16;                                                           T  00470
        GO TO T70;                                                           T  00480
```

```
T30:    IF SUBSTR(STMT, 8, 2)='*2' THEN DO;                                    T  00490
          CODE = '3';                                                          T  00500
          STRT = 10;                                                           T  00510
          GO TO T70;                                                           T  00520
          END;                                                                 T  00530
        CODE = '4';                                                            T  00540
        IF SUBSTR(STMT, 8, 2)='*4'                                             T  00550
          THEN STRT = 10;                                                      T  00560
          ELSE STRT = 8;                                                       T  00570
        GO TO T70;                                                             T  00580
T40:    IF SUBSTR(STMT, 8, 2)='*1' THEN DO;                                    T  00590
          CODE = '7';                                                          T  00600
          STRT = 10;                                                           T  00610
          GO TO T70;                                                           T  00620
          END;                                                                 T  00630
        CODE = '8';                                                            T  00640
        IF SUBSTR(STMT, 8, 2)='*4'                                             T  00650
          THEN STRT = 10;                                                      T  00660
          ELSE STRT = 8;                                                       T  00670
        GO TO T70;                                                             T  00680
T50:    CALL NSTDST (1, NSTNO, CODE, STRT);                                    T  00690
        IF ERRFLG='1'B THEN DO;                                                T  00700
          ERRFLG = '0'B;                                                       T  00710
          RETURN;                                                              T  00720
          END;                                                                 T  00730
        STMT = PLUG(NSTNO) || SUBSTR(STMT, STRT);                              T  00740
        STRT = LENGTH(PLUG(NSTNO)) + 1;                                        T  00750
        GO TO T70;                                                             T  00760
T60:    IF SUBSTR(STMT, 5, 2)='*8' THEN DO;                                    T  00770
          CODE = '2';                                                          T  00780
          STRT = 7;                                                            T  00790
          GO TO T70;                                                           T  00800
          END;                                                                 T  00810
        CODE = '1';                                                            T  00820
        IF SUBSTR(STMT, 5, 2)='*4'                                             T  00830
          THEN STRT = 7;                                                       T  00840
          ELSE STRT = 5;                                                       T  00850
T70:    IF INDEX(ALPHBET, SUBSTR(STMT, STRT, 1))=0 THEN DO;                    T  00860
          CALL ERROR ('T -01: 1ST LTR OF VAR NAME MUST BE A-Z,$.');            T  00870
          RETURN;                                                              T  00880
          END;                                                                 T  00890
        NXTDLM = 1321;                                                         T  00900
        DO SYMB='*','(','/',',',';';                                           T  00910
          SYMLOC = INDEX(SUBSTR(STMT, STRT), SYMB);                            T  00920
          IF SYMLOC=0                                                          T  00930
            THEN SYMLOC = 1321;                                                T  00940
            ELSE SYMLOC = SYMLOC + STRT - 1;                                   T  00950
          IF SYMLOC<NXTDLM THEN NXTDLM = SYMLOC;                               T  00960
```

```
    END;                                                                    T  00970
  SYMB = SUBSTR(STMT, NXTDLM, 1);                                           T  00980
  IF NXTDLM-STRT>6 THEN DO;                                                 T  00990
    CALL ERROR ('T -02: VAR NAME HAS MORE THAN SIX CHAR.');                 T  01000
    RETURN;                                                                 T  01010
    END;                                                                    T  01020
  IF SYMB='*'                                                               T  01030
    THEN DO;                                                                T  01040
      IF CHAR1='N' THEN DO;                                                 T  01050
        CALL ERROR ('T -03: ILLEGAL ''*'' IN EXP NONSTD SPEC.');            T  01060
        RETURN;                                                             T  01070
        END;                                                                T  01080
      IF CHAR1='C' THEN IF SUBSTR(STMT, NXTDLM+1, 2)='16'                   T  01090
        THEN TCODE = '6';                                                   T  01100
        ELSE TCODE = '5';                                                   T  01110
      IF CHAR1='I' THEN IF SUBSTR(STMT, NXTDLM+1, 1)='2'                    T  01120
        THEN TCODE = '3';                                                   T  01130
        ELSE TCODE = '4';                                                   T  01140
      IF CHAR1='L' THEN IF SUBSTR(STMT, NXTDLM+1, 1)='1'                    T  01150
        THEN TCODE = '7';                                                   T  01160
        ELSE TCODE = '8';                                                   T  01170
      IF CHAR1='R' THEN IF SUBSTR(STMT, NXTDLM+1, 1)='8'                    T  01180
        THEN TCODE = '2';                                                   T  01190
        ELSE TCODE = '1';                                                   T  01200
      END;                                                                  T  01210
    ELSE TCODE = CODE;                                                      T  01220
  IF LENGTH(EXPLIST)>1991 THEN DO;                                          T  01230
    CALL ERROR ('T -04: DESIGN PARAMETER EXCEEDED: EXPLIST.');              T  01240
    RETURN;                                                                 T  01250
    END;                                                                    T  01260
  EXPLIST = EXPLIST || SUBSTR(STMT, STRT, NXTDLM-STRT) || '(' ||            T  01270
            TCODE || ')';                                                   T  01280
  IF CHAR1='N' THEN DO;                                                     T  01290
    IF LENGTH(EXPNSL)>993 THEN DO;                                          T  01300
      CALL ERROR ('T -05: DESIGN PARAMETER EXCEEDED: EXPNSL.');             T  01310
      RETURN;                                                               T  01320
      END;                                                                  T  01330
    EXPNSL = EXPNSL || SUBSTR(STMT, STRT, NXTDLM-STRT) || '|';              T  01340
    IF SYMB='('                                                             T  01350
      THEN IF SUBSTR(STMT, NXTDLM, 3)='(*)'                                 T  01360
        THEN DO;                                                            T  01370
          IF LENGTH(NSFNS)>134 THEN DO;                                     T  01380
            CALL ERROR ('T -06: DESIGN PARM EXCEEDED: NSFNS.');             T  01390
            RETURN;                                                         T  01400
            END;                                                            T  01410
          NSFNS = NSFNS || SUBSTR(STMT, STRT, NXTDLM-STRT) ||'|';           T  01420
          IF SUBSTR(STMT, STRT-1, 1)=','                                    T  01430
            THEN DO;                                                        T  01440
```

```
                  STMT = SUBSTR(STMT, 1, STRT-2) || SUBSTR(STMT,      T  01450
                         NXTDLM+3);                                   T  01460
                  NXTDLM = STRT - 1;                                  T  01470
                  END;                                                T  01480
                ELSE DO;                                              T  01490
                  STMT = SUBSTR(STMT, 1, STRT-1) || SUBSTR(STMT,      T  01500
                         NXTDLM+3);                                   T  01510
                  IF SUBSTR(STMT, STRT, 1)=';' THEN RETURN;           T  01520
                  NXTDLM = STRT;                                      T  01530
                  END;                                                T  01540
              SYMB = SUBSTR(STMT, NXTDLM, 1);                         T  01550
              IF INDEX(',;', SYMB)=0 THEN DO;                         T  01560
                CALL ERROR ('T -07: ILLEGAL CHAR AFTER NONSTD FN.');  T  01570
                RETURN;                                               T  01580
                END;                                                  T  01590
              END;                                                    T  01600
            ELSE STMT = SUBSTR(STMT, 1, NXTDLM) || NSDIM(NSTNO) ||    T  01610
                        ',' || SUBSTR(STMT, NXTDLM+1);                T  01620
          ELSE DO;                                                    T  01630
            STMT = SUBSTR(STMT, 1, NXTDLM-1) || '(' || NSDIM(NSTNO)   T  01640
                   || ')' || SUBSTR(STMT, NXTDLM);                    T  01650
            SYMB = '(';                                               T  01660
            END;                                                      T  01670
        END;                                                          T  01680
      IF SYMB='*' THEN DO;                                            T  01690
        IF SUBSTR(STMT, NXTDLM+1, 2)='16'                             T  01700
          THEN NXTDLM = NXTDLM + 3;                                   T  01710
          ELSE NXTDLM = NXTDLM + 2;                                   T  01720
        SYMB = SUBSTR(STMT, NXTDLM, 1);                               T  01730
        END;                                                          T  01740
      IF SYMB='(' THEN DO;                                            T  01750
        IF LENGTH(DIMLIST)>993 THEN DO;                               T  01760
          CALL ERROR ('T -08: DESIGN PARAMETER EXCEEDED: DIMLIST.');  T  01770
          RETURN;                                                     T  01780
          END;                                                        T  01790
        DIMLIST = DIMLIST || SUBSTR(STMT, STRT, NXTDLM-STRT) || '|';  T  01800
        CLPR = INDEX(SUBSTR(STMT, NXTDLM), ')') + NXTDLM - 1;         T  01810
        IF CLPR=NXTDLM-1 THEN DO;                                     T  01820
          CALL ERROR ('T -09: NO CLOSE PAREN FOUND.');                T  01830
          RETURN;                                                     T  01840
          END;                                                        T  01850
        NXTDLM = CLPR + 1;                                            T  01860
        SYMB = SUBSTR(STMT, NXTDLM, 1);                               T  01870
        END;                                                          T  01880
      IF SYMB='/' THEN DO;                                            T  01890
        CLSLSH = INDEX(SUBSTR(STMT, NXTDLM+1), '/') + NXTDLM;         T  01900
        IF CLSLSH=NXTDLM THEN DO;                                     T  01910
          CALL ERROR ('T -10: NO CLOSE SLASH FOUND.');                T  01920
```

```
            RETURN;                                                          T  01930
            END;                                                             T  01940
          NXTDLM = CLSLSH + 1;                                               T  01950
          SYMB = SUBSTR(STMT, NXTDLM, 1);                                    T  01960
          END;                                                               T  01970
        IF SYMB=',' THEN DO;                                                 T  01980
          IF SUBSTR(STMT, NXTDLM-1, 1)=' '                                   T  01990
            THEN DO;                                                         T  02000
              STMT = SUBSTR(STMT, 1, NXTDLM-1) || SUBSTR(STMT,               T  02010
                     NXTDLM+1);                                              T  02020
              STRT = NXTDLM;                                                 T  02030
              END;                                                           T  02040
            ELSE STRT = NXTDLM + 1;                                          T  02050
          GO TO T70;                                                         T  02060
          END;                                                               T  02070
        IF SYMB=';' THEN DO;                                                 T  02080
          IF CHAR1='N'                                                       T  02090
            THEN DO;                                                         T  02100
              STMT = SUBSTR(STMT, 1, NXTDLM-1);                              T  02110
              CALL NUCARD;                                                   T  02120
              END;                                                           T  02130
            ELSE CALL DUPEIT;                                                T  02140
          RETURN;                                                            T  02150
          END;                                                               T  02160
        CALL ERROR ('T -11: ILLEGAL CHAR FOLLOWS PAREN OR SLASH.');          T  02170
T80:    RETURN;                                                              T  02180
        END TYPSTMT;                                                         T  02190
```

## APPENDIX D

## SYNTAX RULES FOR NSFORT STATEMENTS

There are only three statements in NSFORT that are different from normal FORTRAN. The syntax rules for these statements are listed below. The syntax rules for regular FORTRAN statements are in accordance with Reference 8.

Common to the three NSFORT statements is the word NONSTANDARDi*n, which may at any time be abbreviated to NONSTDi*n or to NSTDi*n. The rule for this word is as follows

$$\left\{\begin{array}{l}\text{NONSTANDARD}\\ \text{NONSTD}\\ \text{NSTD}\end{array}\right\} \text{i*n}$$

where i -- is a number from 1 through 6 that signifies the FORTRAN variable type to be mapped into. The numbers 1 through 6 represent respectively REAL*4, REAL*8, INTEGER*2, INTEGER*4, COMPLEX*8, and COMPLEX*16. There is no mapping to LOGICAL variables.

n -- is a number from 1 through 999 and represents the size of the nonstandard variable. Each nonstandard variable will map into a FORTRAN array whose first dimension is n.

### IMPLICIT Statement

The syntax rules for the IMPLICIT Statement are the same as for FORTRAN except that NONSTANDARDi*n (or an accepted

abbreviation) may be used in the same manner as COMPLEX, INTEGER, LOGICAL, and REAL.

NONSTANDARDi*n Statement.

The syntax rules for the nonstandard explicit specification statement are as follows

$$\left\{\begin{matrix}\text{NONSTANDARD}\\ \text{NONSTD}\\ \text{NSTD}\end{matrix}\right\} \text{i*n } a_1(s_1)/x_1/,a_2(s_2)/x_2/,\ldots, a_j(s_j)/x_j/,\ldots,a_k(s_k)/x_k/$$

where $a_j$ -- is a nonstandard variable, array, or function name. If $a_j$ is a nonstandard function name then $(s_j)$ must be (*) and $/x_j/$ must not be used.

$(s_j)$ -- is optional and gives dimension information for arrays or, if $(s_j)$ is (*), identifies nonstandard function names. Each $s_j$ is a series of up through 6 integer constants or integer variables if in a subprogram, separated by commas, which represent the maximum dimensions of the array.

$(x_j)$ -- is optional and represents initial data values. The number and order of the data values must account for the later addition, by the NSFORT translator, of n as the first dimension.

NONSTANDARDi*n FUNCTION Statement

The syntax rules for the NONSTANDARDi*n FUNCTION Statement are as follows

$$\left\{\begin{matrix}\text{NONSTANDARD}\\\text{NONSTD}\\\text{NSTD}\end{matrix}\right\}\ \text{i*n FUNCTION name } (a_1,a_2,\ldots,a_j,\ldots,a_k)$$

where name -- is the name of the nonstandard function.

$a_j$ -- is an argument. It must be a nonsubscripted variable, array, or dummy name of another subprogram. There must be at least one argument in the argument list.

## APPENDIX E

## SUMMARY OF NSFORT CODES

Below are listed the various codes that are used by the NSFORT translator.

### A. VARIABLE TYPE CODES

| Variable Type | Code | Variable Type | Code |
|---|---|---|---|
| REAL*4 | 1 | NONSTANDARD1 | A |
| REAL*8 | 2 | NONSTANDARD2 | B |
| INTEGER*2 | 3 | NONSTANDARD3 | C |
| INTEGER*4 | 4 | NONSTANDARD4 | D |
| COMPLEX*8 | 5 | NONSTANDARD5 | E |
| COMPLEX*16 | 6 | NONSTANDARD6 | F |
| LOGICAL*1 | 7 | | |
| LOGICAL*4 | 8 | | |

### B. OPERATION CODES

| Operation | Code |
|---|---|
| Addition: + | A |
| Subtraction: - | S |
| Multiplication: * | M |
| Division: / | D |
| Exponentiation: ** | E |
| Unary minus: - | N |
| Compare: .LT. | 1 |
| Compare: .LE. | 2 |
| Compare: .EQ. | 3 |
| Compare: .NE. | 4 |
| Compare: .GE. | 5 |
| Compare: .GT. | 6 |

C. SUBROUTINE NAME CODE

When operations (or comparisons) involving nonstandard variables are to be performed, the translator generates a CALL statement to a subroutine which will perform this task. The subroutine name uniquely identifies this task by means of the following format:

N$abcd

where N$ -- uniquely identifies this subroutine as one which performs tasks involving nonstandard variables.

a -- is the VARIABLE TYPE CODE representing the variable type of the result of the operation or comparison. The result is the first argument in the argument list for this subroutine.

b -- is the VARIABLE TYPE CODE representing the type of the first operand of the operation or comparison. The first operand is the second argument in the argument list.

c -- is the OPERATION CODE for the operation or comparison to be performed. If "c" is not generated then the operation is simple arithmetic assignment. (if "c" is not generated, then "d" is also not generated.)

d -- is the VARIABLE TYPE CODE representing the type of the second operand of the operation or comparison. The second operand is the third argument in the argument list. If "c" is generated and "d" is not, then the operation is necessarily unary minus.

The user must supply the subroutines for which the NSFORT translator generates CALL statements.

D. TEMPORARY VARIABLE NAME CODE

When temporary variables are created by the NSFORT translator the format employed is

T$a(i,j)

where T$ -- uniquely identifies the variable as one created by the NSFORT translator.

a -- is the VARIABLE TYPE CODE for this temporary variable.

i -- is the subscript which represents the size of a nonstandard temporary variable. If the VARIABLE TYPE CODE is 1 through 8 (not nonstandard) this first subscript is not used and the format is T$a(j).

j -- is the subscript which uniquely identifies different temporary values.

The user has no control over temporary variables.

## APPENDIX F

## TRANSLATOR ACTION FOR SPECIFIC FORTRAN AND NSFORT STATEMENTS

### Arithmetic Assignment Statement.

If the assignment statement contains no variables that have been identified as nonstandard then no action is taken. If nonstandard variables are present then the arithmetic expression on the right of the "=" symbol is processed, yielding one or more temporary variables or, in the case of a simple assignment statement (e.g. "A=B"), the actual variable is used. (See Arithmetic or Logical Expression below.) A CALL statement is then generated to effect the actual assignment. Implicitly declared nonstandard variables encountered in this statement are dimensioned with the nonstandard size if this action has not been previously taken. For example,

```
NSFORT:        IMPLICIT NONSTANDARD1*2(A-C),LOGICAL(P-R)
             1 A=B+C
             2 A=100.
             3 P=B.GT.C

FORTRAN:       IMPLICIT REAL*4 (A-C),LOGICAL(P-R)
               REAL*4 T$A(2,1)
               LOGICAL*1 T$7(1)
               DIMENSION C(2),B(2),A(2)
             1 CALL N$AAAA (T$A(1,1), B(1), C(1))
               CALL N$AA   (A(1), T$A(1,1))
             2 CALL N$A1   (A(1), 100.)
             3 CALL N$7A6A (T$7(1), B(1), C(1))
               P = T$7(1)
```

Arithmetic or Logical Expression.

Translator action is contingent on the type of statement containing the expression. Using the common reverse Polish and triplet method [Ref. 9] the expression is reduced to a series of binary operations (except in the cases of unary minus and ".NOT."). Nonstandard function references, when encountered, are modified to call a subroutine in a manner compatible with the NSFORT translator's processing of NONSTANDARDi*n-FUNCTION statements. The subroutine name is formed by truncating the function name to 5 characters and concatenating a "$" symbol. (See example below.) For each triplet containing nonstandard variables a CALL statement is constructed which will cause the desired operation or comparison to be performed. For triplets not containing nonstandard variables, a FORTRAN assignment statement is constructed which will perform the desired operation or comparison. In both of the above cases the type of the result is determined by the NSFORT translator based on the types of the operand(s). Temporary variables of the appropriate type are used to store values of intermediate triplets. The temporary variable containing the result of the final triplet contains the value of the expression. If the expression was initially degenerate (i.e. a single variable name) then no temporary variables are used. Implicitly declared nonstandard variables encountered in an expression are dimensioned with the nonstandard size if this action has not been previously taken. For example,

```
NSFORT:         IMPLICIT NONSTANDARD2*3(A-C),LOGICAL(P-R)
                NONSTANDARD2*3 AVG123(*)
              1 A100=AVG123(B100,B101,B102,0.)**2
              2 P=B100.LE.0..AND.B101.LE.0.

FORTRAN:        IMPLICIT REAL*8 (A-C),LOGICAL(P-R)
                REAL*8 T$B(3,2)
                LOGICAL*1 T$7(3)
                DIMENSION B102(3),B101(3),B100(3),A100(3)
              1 CALL $AVG12 (T$B(1,1), B100(1), B101(1),
               *B102(1), 0.)
                CALL N$BBE4 (T$B(1,2), T$B(1,1), 2)
                CALL N$BB   (A100(1), T$B(1,2))
              2 CALL N$7B21 (T$7(1), B100(1), 0.)
                CALL N$7B21 (T$7(2), B101(1), 0.)
                T$7(3) = T$7(1) .AND. T$7(2)
                P = T$7(3)
```

Arithmetic IF Statement.

If the expression in the arithmetic IF statement contains no nonstandard variables then no action is taken. If nonstandard variables are present the expression is processed. (See Arithmetic or Logical Expression.) The resulting expression in this case will necessarily be nonstandard. A CALL statement will then be constructed which will cause a ".LT." comparison of the expression result with zero. Next, a logical IF statement will be constructed which, if the above comparison is true, will cause control to be passed to the first statement number in the original arithmetic IF statement. Similarly, another CALL statement will be constructed which will cause an ".EQ." comparison of the expression result with zero. Again, a logical IF statement will be constructed which, if the preceding comparison is true, will cause control to be

passed to the second statement number. Following this a GO TO statement will be constructed which will by default pass control to the third statement number. For example

```
NSFORT:         IMPLICIT NONSTANDARD4*2(A-C)
              1 IF (A100+A101) 10,20,30

FORTRAN:        IMPLICIT INTEGER*4 (A-C)
                INTEGER*4 T$D(2,1)
                LOGICAL*1 T$7(1)
                DIMENSION A101(2),A100(2)
              1 CALL N$DDAD (T$D(1,1), A100(1), A101(1))
                CALL N$7D11 (T$7(1), T$D(1,1), 0.0)
                IF (T$7(1)) GO TO 10
                CALL N$7D31 (T$7(1), T$D(1,1), 0.0)
                IF (T$7(1)) GO TO 20
                GO TO 30
```

ASSIGN Statement.

No action is taken.

Assigned GO TO Statement.

No action is taken.

Assignment Statement.

See Arithmetic Assignment Statement or Logical Assignment Statement.

AT Statement.

No action is taken.

BACKSPACE Statement.

No action is taken.

BLOCK DATA Statement.

No action is taken.

CALL Statement.

If the argument list contains no nonstandard variables then no action is taken. If nonstandard variables are

present in the argument list then each argument that contains a nonstandard variable is individually processed. (See Arithmetic or Logical Expression above.) The CALL statement is then rewritten with the various expression results replacing those arguments that contained nonstandard variables. For example,

```
NSFORT:        IMPLICIT NONSTANDARD6*3(A-C)
             1 CALL SUBA ('PROB1',A100+A101,B100)

FORTRAN:       IMPLICIT COMPLEX*16 (A-C)
               COMPLEX*16 T$F(3,1)
               DIMENSION A101(3),A100(3),B100(3)
             1 CALL N$FFAF (T$F(1,1), A100(1), A101(1))
               CALL SUBA ('PROB1', T$F(1,1), B100(1))
```

COMMON Statement.

If the COMMON statement contains no nonstandard variables then no action is taken. For each nonstandard variable in the COMMON statement that contains dimension information the nonstandard size is added as the first subscript. For nonstandard variables containing no dimension information and for all non-nonstandard variables no action is taken. Implicitly declared nonstandard variables encountered in this statement are dimensioned (not in this statement) with the nonstandard size if this action has not been previously taken. For example,

```
NSFORT:        IMPLICIT NONSTANDARD2*3(A-C)
               DIMENSION A100(10),A101(10),D100(10)
               COMMON A10C,A101(10),D100(10),A102
```

```
FORTRAN:        IMPLICIT REAL*8 (A-C)
                DIMENSION A102(3)
                DIMENSION A100(3,10),A101(3,10),D100(10)
                COMMONA100,A101(3,10),D100(10),A102
```

COMPLEX Statement.

See Explicit Specification Statement.

COMPLEX FUNCTION Statement.

See FUNCTION Statement.

CONTINUE Statement.

No action is taken.

Control Statements.

See specific control statement.

DATA Initialization Statement.

If the DATA statement contains any implicitly declared nonstandard variables these are dimensioned with the nonstandard size if this action has not been previously taken. For example,

```
NSFORT:         IMPLICIT NONSTANDARD3*2(A-C)
                DATA A100/2*0/

FORTRAN:        IMPLICIT INTEGER*2 (A-C)
                DIMENSION A100(2)
                DATA A100/2*0/
```

See 18, Appendix H for coding restrictions.

DEBUG Statement.

No action is taken.

DEFINE FILE Statement.

No action is taken.

DIMENSION Statement.

The NSFORT translator maintains a list of all dimensioned variable names in order to distinguish between assignment statements and statement functions. The nonstandard size is added as the first subscript for all implicitly declared nonstandard variables. For example,

```
NSFORT:         IMPLICIT NONSTANDARD1*3(A-C)
                DIMENSION A100(4,10),D100(50)

FORTRAN:        IMPLICIT REAL*4 (A-C)
                DIMENSION A100(3,4,10),D100(50)
```

See 8, Appendix H for coding restrictions.

DISPLAY Statement.

No action is taken.

DO Statement.

In order to permit a statement containing nonstandard variables (which will likely be expanded by the NSFORT translator into several statements) to be the last statement in a DO loop, the statement number defining the loop must be modified. The new number will be 99999 minus the number of statement numbers the NSFORT translator has generated up to this point. For example,

```
NSFORT:         IMPLICIT NONSTANDARD5*3(A-C)
                DO 100 I=1,10
                IF (J) 10,20,100
            100 A=B+C
```

```
FORTRAN:        IMPLICIT COMPLEX*8 (A-C)
                COMPLEX*8 T$E(3,1)
                DIMENSION C(3),B(3),A(3)
                DO 99999 I=1,10
                IF (J) 10,20,100
            100 CALL N$EEAE (T$E(1,1), B(1), C(1))
                CALL N$EE   (A(1), T$E(1,1))
          99999 CONTINUE
```

DOUBLE PRECISION Statement.

See Explicit Specification Statement.

DOUBLE PRECISION FUNCTION Statement.

See FUNCTION Statement.

END Statement.

No alteration of the statement is made. This statement causes the NSFORT translator to be made ready to process a new subprogram.

END FILE Statement.

No action is taken.

ENTRY Statement.

No action is taken.

EQUIVALENCE Statement.

If the EQUIVALENCE statement contains no nonstandard variables then no action is taken. For each nonstandard variable in the EQUIVALENCE statement that contains subscripts a "1" is added as the first subscript. For nonstandard variables containing no subscripts and for all non-nonstandard variables no action is taken. Implicitly declared nonstandard variables encountered in this statement are dimensioned (not in this statement) with the nonstandard size if this action has not been previously taken. For example,

```
NSFORT:     IMPLICIT NONSTANDARD2*4 (A-C)
            DIMENSION A100(10),A101(10),D100(10)
            EQUIVALENCE (A100,A101(5),D100(5),A102)

FORTRAN:    IMPLICIT REAL*8 (A-C)
            DIMENSION A102(4)
            DIMENSION A100(4,10),A101(4,10),D100(10)
            EQUIVALENCE(A100,A101(1,5),D100(5),A102)
```

Explicit Specification Statement.

For COMPLEX, DOUBLE PRECISION, INTEGER, LOGICAL, and REAL type statements no alteration of the statement is made. The NSFORT translator maintains lists of all dimensioned variable names and all explicitly declared variable names. See 2, Appendix H for coding restrictions. For explicit nonstandard declaration see NONSTANDARDi*n Statement.

EXTERNAL Statement.

No action is taken.

FIND Statement.

No action is taken.

FORMAT Statement.

No action is taken. See 19, Appendix H for coding restrictions.

FUNCTION Statement.

For FUNCTION statements in which the type is not explicitly stated no action is taken. For FUNCTION statements that explicitly contain COMPLEX, DOUBLE PRECISION, INTEGER, LOGICAL, or REAL no alteration of the statement is made. The NSFORT translator will add the function name to its list of explicitly declared variable names. For nonstandard functions see NONSTANDARDi*n FUNCTION Statement.

GO TO Statement.

No action is taken.

IF Statement.

See Arithmetic IF Statement or Logical IF Statement.

IMPLICIT Statement.

For that portion of the IMPLICIT statement that relates to COMPLEX, INTEGER, LOGICAL, and REAL no alteration of the statement is made. For that portion that makes nonstandard alphabet assignments the IMPLICIT statement is modified in that the NONSTANDARDi*n is replaced by REAL*4, REAL*8, INTEGER*2, INTEGER*4, COMPLEX*8, or COMPLEX*16 as i is 1,2,3,4,5 or 6 respectively. For example,

```
NSFORT:      IMPLICIT INTEGER(P,Q,R), NONSTANDARD1*2 (A-C)

FORTRAN:     IMPLICIT INTEGER(P,Q,R),REAL*4 (A-C)
```

A DIMENSION statement is generated in which all implicitly declared nonstandard variables used in a subprogram are properly dimensioned. See 2, Appendix H for coding restrictions.

Input/Output Statements.

See specific input/output statement.

INTEGER Statement.

See Explicit Specification Statement.

INTEGER FUNCTION Statement.

See FUNCTION Statement.

Logical Assignment Statement.

If the logical assignment statement contains no nonstandard variables then no action is taken. If nonstandard variables are present then the logical expression on the right of the "=" symbol is processed, yielding a temporary logical variable. (See Arithmetic or Logical Expression above.) A logical assignment statement is then constructed to effect the actual assignment. Implicitly declared nonstandard variables encountered are dimensioned with the nonstandard-size if this action has not been previously taken. For example

```
NSFORT:         IMPLICIT NONSTANDARD1*2 (A-C), LOGICAL(P-R)
              1 P=A.GE.B

FORTRAN:        IMPLICIT REAL*4 (A-C),LOGICAL(P-R)
                LOGICAL*1 I$7(1)
                DIMENSION B(2),A(2)
              1 CALL N$7A5A (T$7(1), A(1), B(1))
                P = T$7(1)
```

LOGICAL Statement.

See Explicit Specification Statement.

LOGICAL FUNCTION Statement.

See FUNCTION Statement.

Logical IF Statement.

If neither the logical expression nor the conditional statement contain nonstandard variables then no action is taken. If the logical expression contains nonstandard variables it is processed yielding a temporary logical variable (See Arithmetic or Logical Expression above.)

If the logical expression does not contain nonstandard variables then it is left intact. If the conditional statement does not contain nonstandard variables then the logical IF statement is modified in that the logical expression is replaced by the temporary logical variable. If the conditional statement contains nonstandard variables then control is routed through or around this conditional statement based on the value of the temporary logical variable or the logical expression. For example

```
NSFORT:          IMPLICIT NONSTANDARD3*2 (A-C)
                1 IF (I.EQ.0) GO TO 30
                2 IF (A100.GT.B100) GO TO 40
                3 IF (I.EQ.0) A100=B100+C100
                4 IF (A100.LT.0) A100=-A100

FORTRAN:         IMPLICIT INTEGER*2 (A-C)
                 INTEGER*2 T$C(2,1)
                 LOGICAL*1 T$7(1)
                 DIMENSION B100(2),A100(2),C100(2)
                1 IF (I.EQ.0) GO TO 30
                2 CALL N$7C6C (T$7(1), A100(1), B100(1))
                  IF (T$7(1)) GO TO 40
                3 IF (.NOT.(I.EQ.0)) GO TO 99999
                  CALL N$CCAC (T$C(1,1), B100(1), C100(1))
                  CALL N$CC   (A100(1), T$C(1,1))
            99999 CONTINUE
                4 CALL N$7C14 (T$7(1), A100(1), 0)
                  IF (.NOT.(T$7(1))) GO TO 99998
                  CALL N$CCN  (T$C(1,1), A100(1))
                  CALL N$CC   (A100(1), T$C(1,1))
            99998 CONTINUE
```

NAMELIST Statement.

If the NAMELIST Statement contains any implicitly declared nonstandard variables these are dimensioned with the nonstandard size if this action has not been previously taken. For example

```
NSFORT:        IMPLICIT NONSTANDARD6*3 (A-C)
               NAMELIST /XRAY/ A100,D100

FORTRAN:       IMPLICIT COMPLEX*16 (A-C)
               DIMENSION A100(3)
               NAMELIST /XRAY/ A100,D100
```

NONSTANDARDi*n Statement.

The NONSTANDARDi*n is replaced in the same manner as the IMPLICIT NONSTANDARDi*n. See IMPLICIT Statement. The nonstandard size "n" is added as the first subscript for each variable name in the variable name list. The NSFORT translator maintains a list of all explicitly declared nonstandard variable names. Nonstandard function names that are declared in the NONSTANDARDi*n statement do not appear in the modified type statement. For example

```
NSFORT:        NONSTANDARD4*2 A100,ABCD(*),B100(10,10)/200*0/

FORTRAN:       INTEGER*4 A100(2),B100(2,10,10)/200*0/
```

See 17, Appendix H for coding restrictions.

NONSTANDARDi*n FUNCTION Statement.

"NONSTANDARDi*n FUNCTION" is replaced by "SUBROUTINE," the function name is truncated to 5 characters and concatenated to the "$" symbol, and the untruncated function name is added as the first argument and is explicitly declared and dimensioned based on "i." See IMPLICIT Statement. (Note that this matches with the treatment of function references described in Arithmetic and Logical Expression above.) For example,

```
NSFORT:        NONSTANDARD1*2 FUNCTION SQROOT (A)

FORTRAN:       SUBROUTINE $SQROO (SQROOT,A)
               REAL*4 SQROOT(2)
```

NONSTDI*n Statement.

Accepted abbreviation for NONSTANDARDi*n.

See NONSTANDARDi*n Statement.

NONSTDi*n FUNCTION Statement.

Accepted abbreviation for NONSTANDARDi*n FUNCTION.

See NONSTANDARDi*n FUNCTION Statement.

NSTDi*n Statement.

Accepted abbreviation for NONSTANDARDi*n.

See NONSTANDARDi*n Statement.

NSTDi*n FUNCTION Statement.

Accepted abbreviation for NONSTANDARDi*n FUNCTION Statement. See NONSTANDARDi*n FUNCTION Statement.

Output Statements.

See specific output statement.

PAUSE Statement.

No action is taken.

PRINT Statement.

No action is taken.

PUNCH Statement.

No action is taken.

READ Statement.

If the READ statement contains any implicitly declared nonstandard variables these are dimensioned with the

nonstandard size if this action has not been previously taken. For example,

```
NSFORT:     IMPLICIT NONSTANDARD2*3 (A-C)
            READ (6,10) A100

FORTRAN:    IMPLICIT REAL*8 (A-C)
            DIMENSION A100(3)
            READ (6,10) A100
```

See 19, Appendix H for coding restrictions.

REAL Statement.

See Explicit Specification Statement.

REAL FUNCTION Statement.

See FUNCTION Statement.

RETURN Statement.

No action is taken.

REWIND Statement.

No action is taken.

Specification Statements.

See specific specification statement.

Statement Function Definition Statement.

If the statement function contains no nonstandard variables then no action is taken; otherwise an error message is generated by the NSFORT translator, as statement functions may not contain nonstandard variables. See 10, Appendix H.

STOP Statement.

No action is taken.

Subprogram Statements.

See specific subprogram statement.

SUBROUTINE Statement.

No action is taken.

TRACE Statement.

No action is taken.

Type Statement.

See Explicit Specification Statement or NONSTANDARDi*n Statement.

WRITE Statement.

No action is taken. See 19, Appendix H for coding restrictions.

## APPENDIX G

## RESULT TYPE OF MIXED TYPE OPERATIONS

When an operation involving operands of different variable types is to be performed, the variable type of the result is determined by the table below. See Appendix E for variable type codes.

| Operand Type - Operand B | Operand Type - Operand A | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | A | B | C | D | E | F |
| 1 | 1 | 2 | 1 | 1 | 5 | 6 | A | B | C | D | E | F |
| 2 | 2 | 2 | 2 | 2 | 6 | 6 | A | B | C | D | E | F |
| 3 | 1 | 2 | 3 | 4 | 5 | 6 | A | B | C | D | E | F |
| 4 | 1 | 2 | 4 | 4 | 5 | 6 | A | B | C | D | E | F |
| 5 | 5 | 6 | 5 | 5 | 5 | 6 | A | B | C | D | E | F |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | A | B | C | D | E | F |
| A | A | A | A | A | A | A | A | B | A | A | E | F |
| B | B | B | B | B | B | B | B | B | B | B | E | F |
| C | C | C | C | C | C | C | A | B | C | D | E | F |
| D | D | D | D | D | D | D | A | B | D | D | E | F |
| E | E | E | E | E | E | E | E | E | E | E | E | F |
| F | F | F | F | F | F | F | F | F | F | F | F | F |

## APPENDIX H

## DESIGN LIMITATIONS AND RESTRICTIONS

This appendix lists the limitations in the design of the NSFORT translator and the restrictions to FORTRAN programming which must be observed when using the NSFORT translator. If any of these limitations and restrictions are exceeded or violated then three things may happen: (1) the NSFORT translator will issue an error message, and/or (2) an error will be caused which will be diagnosed by the FORTRAN compiler, or (3) an error will be caused which may produce unintended results. For each of the limitations and restrictions below, a note will indicate whether or not the NSFORT translator will recognize the violation.

1. Use of "$" Symbol. The NSFORT translator uses the "$" symbol to uniquely identify translator generated temporary variable names and subroutine names. Any use of the combination "T$" as the first two characters of a variable or array name is not permitted and may lead to erroneous results. Any use of "N$" or "$" as the first characters of a subroutine name may also cause confusion. (No NSFORT error message.)

2. IMPLICIT and Type Statements. The IMPLICIT statement, if used, must be the first statement in a main program or the second statement in a subprogram. All type statements must follow the IMPLICIT statement and precede all other specification statements. (No NSFORT error message.)

3. Nonstandard Size. Within a subprogram the size for each particular nonstandard type must be consistent. For example, NONSTANDARDi*2 and NONSTANDARDj*3 cannot both be used in the same subprogram when i=j. (Error message generated.)

4. Explicitly Declared Variables. The storage capacity constraint for explicitly declared variable and array names of any type within a subprogram (including nonstandard) is $1 + \Sigma_{i=1}^{n} (x_i + 3) \leq 2000$ where $x_i$ is the number of characters in the $i^{th}$ explicitly declared variable or array name, and n is the total number of explicitly declared variable or array names within a subprogram. (Error message generated.)

5. Explicitly Declared Nonstandard Variables. The storage capacity constraint for explicitly declared nonstandard variable and array names within a subprogram is $1 + \Sigma_{i=1}^{n} (x_i + 1) \leq 1000$ where $x_i$ is the number of characters in the $i^{th}$ explicitly declared nonstandard variable or array name, and n is the total number of explicitly declared nonstandard variable or array names within a subprogram. (Error message generated.)

6. Nonstandard Functions. The storage capacity constraint for nonstandard function names within a subprogram is $1 + \Sigma_{i=1}^{n} (x_i + 1) \leq 141$ where $x_i$ is the number of characters in the $i^{th}$ nonstandard function name, and n is the total number of nonstandard function names within a subprogram. (Error message generated.)

7. Arrays and Nonstandard Variables. The storage capacity constraint for array names and nonstandard variable or array names within a subprogram is $1 + \Sigma_{i=1}^{n} (x_i + 1) \leq 1000$ where $x_i$ is the number of characters in the $i^{th}$ array name or nonstandard variable or array name, and n is the total number of array names and nonstandard variable or array names within a subprogram. (Error message generated.)

8. Explicitly Declared Nonstandard Variables and the DIMENSION Statement. Explicitly declared nonstandard variables which are to be nonstandard arrays must have the dimension information in the NONSTANDARDi*n statement. Explicitly declared nonstandard variable names may not appear in a DIMENSION statement. (Error message generated.)

9. FORTRAN Functions. FORTRAN functions may not contain nonstandard variables in their argument list. (Error message generated.)

10. Statement Functions. Statement functions may not contain nonstandard variables. (Error message generated.)

11. Nonstandard Function Names. Since the NSFORT translator truncates nonstandard function names to 5 characters and concatenates them to a "$" symbol, the first five characters of a nonstandard function name must uniquely identify it. (No NSFORT error message.)

12. Statement Numbers. Within a subprogram the NSFORT translator uses statement numbers beginning with 99999 and continuing downward in decrements of one for as many as

needed. Statement numbers that are in the range used by the translator may not be used. (No NSFORT error message.)

13. Variable Names. Assignment statements that begin with the character sequences "CALL", "DATA", "ENTRY", "FORMAT(", "IF(", "REAL", "READ(", or "WRITE(", excluding blanks, will be mishandled by the NSFORT translator (No NSFORT error message.)

14. Subscripts. Subscripts may not contain nonstandard variables. (Error message generated.)

15. Subscripts. Nonstandard variables may not contain in excess of six subscripts. (No NSFORT error message.)

16. Operands. The combined length of a variable name and all its subscript information or a non-nonstandard function name and is argument list (including commas and parentheses but excluding blanks) in any statement containing nonstandard variables must not exceed 30 characters. For nonstandard variables the limit is 28 as the NSFORT translator must insert two characters. (Error message generated.)

17. Initial Data Values in the NONSTANDARDi*n Statement. The NSFORT translator takes no action on initial data values in the NONSTANDARDi*n statement. It is necessary for the user to insure that the resulting statement after translation will assign values as desired. For example, in the statement "NONSTANDARD1*2 ALPHA(10,10)/$x_n$/", there must be 200 initial values, $x_n$, that will map in the ALPHA array as

desired. The resulting statement will be "REAL*4 ALPHA(2,10,10)/$x_n$/" where $x_n$ is unchanged. (No NSFORT error message.)

18. DATA Statement. The DATA Statement is not transformed by the NSFORT translator and thus caution must be exercised in using nonstandard variables in this statement. Implicitly declared nonstandard variables encountered for the first time will be dimensioned with the nonstandard size, however neither the subscripts nor the number of data values in a DATA statement will be altered. A FORTRAN compiler error will result if a nonstandard variable appears with subscripts because the NSFORT translator will not add a subscript for the nonstandard size. The user may make use of the DATA statement by not subscripting nonstandard variables or by adding the first subscript himself. (No NSFORT error message.)

19. Input/Output. The NSFORT translator takes no action on input/output statements except to dimension with the nonstandard size, implicitly declared nonstandard variables encountered in NAMELIST or READ statements for the first time. Therefore the user will have to add the first subscript for nonstandard variables in which other subscripts are explicitly used in input/output statements. The FORMAT statement must be written by the user to account for the size of nonstandard variables. (No NSFORT error message.)

20. Expressions. The NSFORT translator evaluates expressions using a reverse Polish and triplet method whereby each time a triplet is to enter the reverse Polish string it is replaced by the temporary variable containing the triplet result. In this process there are four storage constraints which cannot be exceeded. The constraint for the length of the reverse Polish string is $\Sigma_{i=1}^{n} (x_i + 1) \leq 100$ where $x_i$ is the length of the $i^{th}$ operand (including subscripts or arguments but excluding blanks) and n is the number of operands present at one time in the reverse Polish string. The constraint for the operator stack is $n \leq 30$ where n is the number of operators or nonstandard functions in the stack at one time. The constraint for the stack of nonstandard function names is $\Sigma_{i=1}^{n} (x_i + 1) \leq 71$ where $x_i$ is the number of characters in the $i^{th}$ nonstandard function name, and n is the number of nonstandard function names in the stack at one time. The constraint for the length of the argument list for a nonstandard function is 100 characters, excluding blanks, after each expression in the argument list has been replaced by a temporary variable. (Error messages generated.)

21. Nested DO Loops. The constraint on nested DO loops is $\Sigma_{i=1}^{n} (x_i + 6) \leq 99$ where $x_i$ is the number of characters in the $i^{th}$ statement number, and n is the number of nested DO loops at any point in the program with different terminal statements. (Error message generated.)

## APPENDIX I

## NSFORT DIAGNOSTIC MESSAGES

The following is a list of all of the NSFORT translator diagnostic messages. These messages are generated when the translator encounters a construction which it does not recognize, when a logic or syntax error is found that prevents proper translation, or when storage space for the translator's bookkeeping is exhausted. It should be emphasized that the translator will not find all errors (see Appendix H). The letter(s) preceding the dash in the code for the messages is an abbreviation for the name of the procedure which found the error. These abbreviations are identified in Appendix A.

A -01: DESIGN PARAMETER EXCEEDED: OPERAND.

The variable name with subscripts or the statement function name with dummy arguments to the left of the "=" symbol in an assignment statement or function exceeds 30 characters excluding blanks. (See 16, Appendix H.)

A -02: STMT FN CONTAINS NONSTD.

A statement function contains nonstandard variables. The identifier on the left side of an arithmetic assignment statement is subscripted and this identifier has not been dimensioned. Therefore this statement is interpreted as a statement function

and it contains nonstandard variables. (See 10, Appendix H.)

A - 03: ASGMT STMT HAS LOG VAR ON ONLY 1 SIDE.

An assignment statement was found to have a logical variable on only one side of the "=" symbol.

A -04: DESIGN PARAMETER EXCEEDED: OPERAND.

See A -01. In this case the variable is nonstandard and "1," must be inserted as the first subscript. Therefore, if the original variable name with subscripts contains in excess of 28 characters, when "1," was added it would exceed 30 characters. (See 16, Appendix H.)

CO-01: NO CLOSE SLASH FOUND.

A slash was found in a comma statement delimiting a labeled common name. No corresponding close slash was found in the statement.

CO-02: ILLEGAL CHAR FOLLOWS COMMA OR SLASH.

The character immediately following a comma or a slash is illegal. The program is looking at the character immediately following COMMON, or a comma, or the closing slash on a labeled common name. It is expecting to find a variable name but the first letter is not a member of the alphabet or there is no variable name.

CO-03: VAR IDENTIFIER HAS > 6 CHAR.

An identifier was found to have more than 6 characters. This occurs when there were more than 6

characters from the first character of an identifier to an open parenthesis, a comma, a slash, or to the end of the COMMON statement.

CO-04: NO CLOSE PAREN FOUND.

A variable name was read, a parenthesis was found indicating array information. No corresponding close parenthesis was found in the remainder of the statement.

CO-05: ILLEGAL CHAR FOLLOWS CLOSE PAREN.

The character immediately following a close parenthesis was not recognized. The translator expected to find a comma, a slash, or the end of the COMMON statement.

DA-01: VAR IDENTIFIER HAS > 6 CHAR.

An identifier was found to have more than 6 characters. More than 6 characters from the first character of the identifier to an open parenthesis, a comma, a slash, or to the end of the DATA statement were found. The first variable was expected to begin immediately following the word DATA.

DA-02: NO CLOSE PAREN FOUND.

A parenthesis was found following a variable name indicating subscript information but no corresponding close parenthesis was found in the remainder of the statement.

DA-03: NO CLOSE SLASH FOUND.

A slash was found indicating constant values but

no corresponding close slash was found in the remainder of the statement.

DA-04: DATA STMT SYNTAX ERROR.

The end of statement was encountered unexpectedly or an illegal character followed a close parenthesis, a close slash, or a comma.

DI-01: OPEN PAREN NOT FOUND WHEN EXPECTED.

The translator was looking for an open parenthesis following a variable name. No open parenthesis was found in the remainder of the statement.

DI-02: EXP DCL NONSTD CHAR IN DIMEN STMT.

An explicitly declared nonstandard variable was found in a dimension statement. (See 8, Appendix H.) When a nonstandard variable is explicitly declared, the dimension information must be contained in that explicit declaration and not in a dimension statement later on in the program.

DI-03: DESIGN PARAMETERS EXCEEDED: DIMLIST.

The number and length of items dimensioned in the program exceeds the storage capacity for this list. (See 7, Appendix H.)

DI-04: NO CLOSE PAREN FOUND WHEN EXPECTED.

An open parenthesis was found indicating dimension information but no close parenthesis was found in the remainder of the statement.

DM-01: DESIGN PARAMETER EXCEEDED: DIMLIST.

See DI-03.

DO-01: STMT NUMBER EXCEEDS 5 DIGITS.

The end-of-range statement number in a DO statement exceeds 5 digits. The statement number begins with the first non zero character following the DO and extends up to the first non-numeric character found (excluding blanks).

DO-02: DESIGN PARAMETER EXCEEDED: LOOPNOS.

The number and length of end-of-range statement numbers of nested DO loops exceeds the storage capacity for this list. (See 21, Appendix H.)

EQ-01: EQUIVALENCE STMT SYNTAX ERROR.

The first character of a variable name was expected but this character was not a member of the alphabet. The first variable was expected to begin immediately following the "(" after EQUIVALENCE.

EQ-02: VAR IDENTIFIER HAS > 6 CHAR.

An identifier was found to have more than 6 characters. More than 6 characters from the first character of the identifier to an open parenthesis, a comma, a close parenthesis or to the end of EQUIVALENCE statement were found. The first variable was expected to begin immediately following the "(" after EQUIVALENCE.

EQ-03: NO CLOSE PAREN FOUND.

An open parenthesis was found but no corresponding close parenthesis was found in the remainder of the sentence.

EQ-04: ILLEGAL CHAR FOLLOWS ')' OR AT END.

The character following a closed parenthesis is not recognized or the end of the EQUIVALENCE statement occurs unexpectedly.

EX-01: '.NOT.' IN ILLEGAL POSITION.

The logical operator '.NOT.' was found in an illegal position. It was expected that '.NOT.' would be first in an expression or immediately following an open parenthesis, a comma, '.AND.', or '.OR.'.

EX-02: 1ST CHAR OF EXPRESSION IS ILLEGAL.

The first character of an arithmetic or logical expression is illegal. The translator expected to find a letter in the alphabet, a number, an open parenthesis, a period, or a minus sign.

EX-03: VAR OR FUNC IDENTIFIER HAS > 6 CHAR.

A variable or function identifier was found to contain more than 6 characters. From the first character the variable or function identifier an arithmetic, relational, or logical operator, an open or close parenthesis, or a comma was not found within 6 characters.

EX-04: DESIGN PARAMETER EXCEEDED: FNSTAK.

When processing expressions, nonstandard function names are stacked when found. The number of nested functions exceeds the storage capacity for this list. (See 20, Appendix H.)

EX-05: NONSTD IN SUBSCRPTS OR FUNC ARG.

When processing an expression, an open parenthesis following an identifier was found indicating subscript information or indicating function arguments. Within this open parenthesis and its corresponding close parenthesis, nonstandard variables were found. Nonstandard variables are not permitted in subscripts or possibly in this case, the function was not identified as a nonstandard function. (See 9, 15, Appendix H.)

EX-06: DESIGN PARAMETER EXCEEDED: RPSTR.

When processing an expression, the length of the reverse Polish string exceeded the storage capacity for that string. (See 20, Appendix H.)

EX-07: ILLEGAL CHARS FOLLOWING '.'.

A period was found and the character immediately following the period was examined. If this character was a numeral, the period is treated as a decimal point. If the second character is non-numeric, then the period was interpreted to be the first period in a logical constant. Therefore another period was expected as the fifth or sixth character after the first period indicating a .TRUE. or a .FALSE. This period was not found in the fifth or sixth character following the first period.

EX-08: ILLEGAL CHAR FOLLOWS LOG CONST.

The program expected to find either a comma, a parenthesis, logical .AND., .OR., or the end of the expression following a logical constant. But some other character was found.

EX-09: DESIGN PARAMETER EXCEEDED: RPSTR.

See EX-06.

EX-10: DESIGN PARAMETER EXCEEDED: RPSTR.

See EX-06.

EX-11: NO CLOSE PAREN FOUND.

An open parenthesis was found in a location other than immediately after an identifier and no corresponding close parenthesis was found in the remainder of the statement.

EX-12: DESIGN PARAMETER EXCEEDED: RPSTR.

See EX-06.

EX-13: ILLEGAL CHAR FOLLOWS CMPLX CONSTANT.

The translator program expected to find an arithmetic operator, a close parenthesis, a comma, or end of expression following the close parenthesis delimiting a complex constant.

EX-14: ILLEGAL CHAR FOLLOWS DELIMITER.

The character immediately following an arithmetic, relational, or logical operator, a parenthesis, or a comma constitutes an illegal FORTRAN construction.

EX-15: UNMATCHED CLOSE PAREN FOUND.

A close parenthesis was encountered but it has no corresponding open parenthesis preceding it.

EX-16: DESIGN PARAMETER EXCEEDED: STCK.

When processing expressions by the reverse Polish string method operators are stacked as described in Reference 9. The stack exceeds the storage capacity allowed by the translator. (See 20, Appendix H.)

EX-17: UNMATCHED OPEN PAREN FOUND.

An open parenthesis was encountered but it has no corresponding close parenthesis in the remainder of the statement.

EX-18: DESIGN PARAMETER EXCEEDED: STCK.

See EX-16.

EX-19: DESIGN PARAMETER EXCEEDED: OPERAND.

A variable identifier with subscripts or a non-nonstandard function identifier with arguments exceeds 30 characters excluding blanks. (See 16, Appendix H.)

EX-20: DESIGN PARAMETER EXCEEDED: ARGS.

The argument list for a nonstandard function identifier exceeds 100 characters, excluding blanks, after each expression in the argument list has been replaced by a temporary variable. (See 20, Appendix H.)

EX-21: DESIGN PARAMETER EXCEEDED: RPSTR.

See EX-06.

EX-22: DESIGN PARAMETER EXCEEDED: OPERAND

See EX-19.

EX-23: LOG VAR FOUND IN ARITH EXPR.

In processing an expression a logical variable or constant was encountered but the associated operator was not a logical operator.

EX-24: LOG EXPR CONTAINS NONLOG VAR.

A logical operator was encountered but an associated operand was found which was not a logical variable.

EX-25: DESIGN PARAMETER EXCEEDED: RPSTR.

See EX-06.

EX-26: EXPR SYNTAX IN ERROR.

Upon reaching the end of the expression and emptying the operator stack the reverse Polish string should have been reduced to a single temporary variable. (See Chapter III.) But instead more than one variable was found remaining in the reverse Polish string.

EX-27: DESIGN PARAMETER EXCEEDED: OPERAND.

See EX-19.

EX-28: DESIGN PARAMETER EXCEEDED: OPERAND.

A nonstandard variable identifier with subscripts exceeds 28 characters excluding blanks. (See 16, Appendix H.)

F -01: NO OPEN PAREN FOUND IN FUNCTION STMT.

In a FUCTION statement the open parenthesis that precedes the dummy argument list was not found.

F -02: FUNC IDENTIFIER HAS > 6 CHAR.

In a FUNCTION statement the names of the function was found to have greater than 6 characters.

I -01: IMPLICIT SPEC. MUST DEAL W/A-Z,$.

In an IMPLICIT statement the first character following an open parenthesis or a comma was not an alphabetic character.

I -02: LAST CHAR IN IMP SPEC PRECEDES 1ST.

In an IMPLICIT statement the translator found the construction: q b h c r where q is a comma or open parenthesis, b is an alphabetic character, h is not a comma and not a close parenthesis (expected to be a hyphen), c is a character which is not alphabetic or is alphabetic but precedes b in the alphabetic character list, and r is a comma or close parenthesis.

I -03: UNRECOGNIZABLE STATEMENT.

The statement was classified as IMPLICIT since it was determined not to be an assignment statement and it began with IM, but in further processing one of the following unfamiliar constructions were found.

a. The ninth character in the statement or the second character following a close parenthesis was not a C, I, L, N, or R (COMPLEX, INTEGER, etc.).

b. The character sequence following COMPLEX was not "(", "*8(", or "*16("; or the character sequence following INTEGER was not "(", "*2(", or "*4("; etc.

c. In the alphabet lists with the construction q b r c s where q is a comma or open parenthesis, b is an alphabetic character, and c is any character, a comma or close parenthesis was expected in position r or s but none was found.

IF-01: OPEN AND CLOSE PARENS ADJACENT.

If an IF statement the construction IFa), where a is any character, was found.

IF-02: ARITH IF STMT CONTAINS LOG EXPRSN.

The IF statement was classified as arithmetic but a logical expression was found within the parentheses.

IF-03: 2ND COMMA NOT FOUND IN ARITH IF STMT.

The IF statement was classified as arithmetic because in the right side of the statement a comma was found without parentheses. But in processing the statement the required second comma was not found.

IF-04: DESIGN PARAMETER EXCEEDED: LOOPNOS.

The translator maintains a temporary stack of nested DO loop end-of-range statement numbers. In the processing of a logical IF statement the translator also uses this stack. The storage allocated to this stack is full. (See 21, Appendix H.)

L -01: STMT END PRECEDES LITERAL END.

A literal was found which was identified by the code nH, where n is an integer constant less than 256, but n exceeds the remaining length of the statement.

L -02: NO CLOSE QUOTE WAS FOUND.

A literal was found which was identified by an apostrophe but no corresponding close quote was found. Note that two adjacent apostrophes cannot signify the end of a literal.

M -01: UNEXPECTED CONTINUATION CARD.

A NSFORT source program card (other than a comment card) with a non-zero, non-blank character in column 6 was encountered as the first card in a subprogram, or following a comment card, or following excessive continuation cards. See M -02.

M -02: EXCESSIVE CONTINUATION CARDS OVER 19.

A statement was found to occupy in excess of 20 cards.

NA-01: NO CLOSE SLASH FOUND.

A slash was found in a NAMELIST statement but no corresponding close slash was found in the remainder of the statement.

NA-02: VAR IDENTIFIER HAS > 6 CHAR.

An identifier was found to have more than 6 characters. More than 6 characters from the first character of the identifier to a comma, a slash, or to the end of the NAMELIST statement were found.

NA-03: ILLEGAL CHAR FOLLOWS COMMA OR SLASH.

The character following a comma or slash was expected to be a letter of the alphabet (or $) but another character was found.

NS-01: NONSTANDARD SYNTAX ERROR.

An "*" was expected as the 6th, 8th, or 13th character in the NONSTANDARDi*n phrase, but was not found. (See Appendix D.)

NS-02: NONSTANDARD TYPE IS NOT 1-6.

A character other than 1,2,3,4,5 or 6 was found preceding the "*" in the NONSTANDARDi*n phrase. (See Appendix D.)

NS-03: NONSTANDARD SIZE IS UNRECOGNIZABLE.

A non-numeric character was found immediately following the "*" in the NONSTANDARDi*n phrase. (See Appendix D.)

NS-04: NONSTD SIZE EXCEEDS THREE DIGITS.

The n in the NONSTANDARDi*n phrase was found to exceed three digits. (See Appendix D.)

NS-05: NONSTD-N USED W/DIFFERENT SIZE.

Within a subprogram a particular nonstandard type was used with differing sizes. This inconsistency is not permitted. (See 3, Appendix H.)

P -01: UNRECOGNIZABLE STATEMENT.

The translator was attempting to classify a statement but could not. The statement was determined to be neither an assignment statement nor a statement

function and the first two characters were unrecognizable.

PA-01: UNMATCHED PARENS FOUND.

An open parenthesis was found but no corresponding close parenthesis was located in the remainder of the statement.

R -01: UNFAMILIAR READ STMT SYNTAX.

The translator was searched for the beginning of the input list. If an open parenthesis follows READ then the translator failed to find a close parenthesis.

R -02: ILLEGAL 1ST CHAR IN I/O LIST ITEM.

The first character in the input list was found to be other than an open parenthesis or letter of the alphabet.

R -03: VAR IDENTIFIER HAS > 6 CHAR.

An identifier was found to have in excess of 6 characters. More than 6 characters from the first character of the identifier to an open parenthesis, a comma, or to the end of the READ statement were found.

R -04: READ STMT SYNTAX ERROR.

The translator did not find a comma or the end of the READ statement following a close parenthesis.

T -01: 1ST LTR OF VAR NAME MUST BE A-Z,$.

The first character following the type*n phrase or following a comma is not a letter of the alphabet.

T -02: VAR NAME HAS MORE THAN SIX CHAR.

An identifier was found to have in excess of 6 characters. More than 6 characters from the first character of the identifier to an asterisk, an open parenthesis, a slash, a comma, or to the end of the statement were found.

T -03: ILLEGAL '*' IN EXP NONSTD SPEC.

An asterisk other than that in the NONSTANDARDi*n phrase or those identifying nonstandard function names was found. (See Appendix D.)

T -04: DESIGN PARAMETER EXCEEDED: EXPLIST.

The number and length of explicitly declared variable names exceeds the storage capacity for this list. (See 4, Appendix H.)

T -05: DESIGN PARAMETER EXCEEDED: EXPNSL.

The number and length of explicitly declared nonstandard variable names exceeds the storage capacity for this list. (See 5, Appendix H.)

T -06: DESIGN PARAMETER EXCEEDED: NSFNS.

The number and length of nonstandard function names exceeds the storage capacity for this list. (See 6, Appendix H.)

T -07: ILLEGAL CHAR AFTER NONSTD FN.

Following the declaration of a nonstandard function name, the translator expected to find a comma or the end of the statement.

T -08: DESIGN PARAMETER EXCEEDED: DIMLIST.

See DI-03.

T -09: NO CLOSE PAREN FOUND.

An open parenthesis was encountered but no corresponding close parenthesis was found in the remainder of the statement.

T -10: NO CLOSE SLASH FOUND.

An open slash was found but no corresponding close slash was found in the remainder of the statement.

T -11: ILLEGAL CHAR FOLLOWS PAREN OR SLASH.

The character following "*16", "*_', a close parenthesis, or a close slash is not legal.

LIST OF REFERENCES


1. Wengert, R. E., "A Simple Automatic Derivative Evaluation Program," Communications of the ACM, v. 7, p. 463-464, n. 8 (August, 1964).

2. Wilkins, R. D., "Investigation of a New Analytical Method for Numerical Derivative Evaluation," Communications of the ACM, v. 7, p. 465-471, n. 8 (August, 1964).

3. Moore, Ramon E., Interval Analysis, Prentice-Hall, 1966.

4. Shudde, R., Automatic Error Analysis Via Range (interval) Arithmetic, program description, Computer Facility, Naval Postgraduate School, January, 1966.

5. Control Data Corporation, FORTRAN 63/ Reference Data, 1604/ 1604-A Computer, Rev. A, June, 1964.

6. International Business Machines, IBM System/360 Operating System, PL/I (F), Language Reference Manual, 3rd ed., October, 1969.

7. International Business Machines, IBM System/360 Operating System, PL/I (F), Programmer's Guide, 5th ed., November, 1968.

8. International Business Machines, IBM System/360, FORTRAN IV Language, 6th ed.

9. Randall, B., and Russell, L. J., ALGOL 60 Implementation, Academic Press, 1964.

INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |
| 2. | Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 3. | Assoc. Professor R. H. Shudde, Code 55 Su<br>Department of Operations Analysis<br>Naval Postgraduate School<br>Monterey, California 93940 | 5 |
| 4. | LCDR E. A. Singer<br>Department of Mathematics<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 5. | Department of Operations Analysis<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 6. | Department of Mathematics<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 7. | W. R. Church Computer Center<br>Naval Postgraduate School<br>Monterey, California 93940 | 5 |
| 8. | Captain David R. Little, USMC<br>112 Blackstone Avenue<br>Ithaca, New York 14850 | 5 |

Blank

## DOCUMENT CONTROL DATA - R & D

*(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)*

1. ORIGINATING ACTIVITY (Corporate author)
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE

NSFORT--A NONSTANDARD FORTRAN LANGUAGE TRANSLATOR

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Master's Thesis; September 1970

5. AUTHOR(S) (First name, middle initial, last name)
David Ross Little, Captain, United States Marine Corps

| 6. REPORT DATE | 7a. TOTAL NO. OF PAGES | 7b. NO. OF REFS |
|---|---|---|
| September 1970 | 154 | 9 |

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

This paper presents a computer language translator which allows the IBM FORTRAN G (or higher level) user to solve directly problems in which the variables may be n-tuples and/or the FORTRAN arithmetic and relational operations may be defined by the user. The translator, called the NSFORT (for NonStandard FORTRAN) translator, will (1) decompose all expressions to a series of binary arithmetic operations, relational operations, or user defined functions; (2) generate CALL statements to user supplied subprograms to perform the above operations; and (3) produce a new source program that is in every respect acceptable to the FORTRAN compiler. While using the NSFORT translator the user has virtually unrestricted use of FORTRAN. The translator's applications include n-precision arithmetic, vector and matrix operations, numerically evaluated analytic derivatives, interval arithmetic, and others. The paper describes completely the use and operation of the translator, provides examples, indicates applications, and discusses programming techniques.

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6811





A-31408

| 4. KEY WORDS | LINK A | | LINK B | | LINK C | |
|---|---|---|---|---|---|---|
| | ROLE | WT | ROLE | WT | ROLE | WT |
| FORTRAN | | | | | | |
| COMPUTER APPLICATIONS | | | | | | |
| INTERVAL ARITHMETIC | | | | | | |

120915

Thesis
L693
c.1

Little
NSFORT--a nonstand-
ard FORTRAN language
translator.